Integer is Enough: When Vertical Federated Learning Meets Rounding

Pengyu Qiu[1,2], Yuwen Pu[1], Yongchao Liu[2*], Wenyan Liu[1,2], Yun Yue[2], Xiaowei Zhu[2], Lichun Li[2], Jinbao Li[3*], Shouling Ji[1*]

[1]Zhejiang University
[2]Ant Group
[3]Qilu University of Technology
qiupys@zju.edu.cn, yw.pu@zju.edu.cn, yongchao.ly@antgroup.com, enxuan.lwy@antgroup.com, yueyun.yy@antgroup.com, robert.zxw@antgroup.com, licun.llc@zmxy.com.cn, Lijinb@sdas.org, sji@zju.edu.cn

Abstract

Vertical Federated Learning (VFL) is a solution increasingly used by companies with the same user group but differing features, enabling them to collaboratively train a machine learning model. VFL ensures that clients exchange intermediate results extracted by their local models, without sharing raw data. However, in practice, VFL encounters several challenges, such as computational and communication overhead, privacy leakage risk, and adversarial attack. Our study reveals that the usage of floating-point (FP) numbers is a common factor causing these issues, as they can be redundant and contain too much information. To address this, we propose a new architecture called rounding layer, which converts intermediate results to integers. Our theoretical analysis and empirical results demonstrate the benefits of the rounding layer in reducing computation and memory overhead, providing privacy protection, preserving model performance, and mitigating adversarial attacks. We hope this paper inspires further research into novel architectures to address practical issues in VFL.

Introduction

Vertical Federated Learning (VFL) (Yang et al. 2019) has become a popular solution for companies with the same group of users but different features, enabling them to collaboratively train a model. In popular frameworks like FATE (Liu et al. 2021), clients submit intermediate results extracted by their local models instead of raw data. The server then aggregates these intermediate results for further computation and returns corresponding gradients for training. Homomorphic Encryption (HE) (Zhang and Zhu 2020; Zhang et al. 2018) is used in the framework to provide the confidentiality of these intermediate results, which enables the computation to be executed in an encrypted form. However, despite the usage of HE, there are several challenges inherent in this design.

Computational Overhead While HE provides privacy protection, it can require significant computational resources for training and inference, particularly when calculations involve floating-point (FP) numbers. Indeed, HE cannot directly apply to calculations involving FP numbers. A naive solution is that FP numbers must be converted to fixed-point numbers for further processing; while other solutions involves complex designs (Moon and Lee 2020). These solutions can consume additional time and resources.

Privacy Leakage While protecting intermediate results from being accessed in plain text is crucial, research has found that there are still privacy leakage risks (Luo et al. 2020; Qiu et al. 2022a). For instance, in a two-party scenario, one curious party can reconstruct the other party's intermediate results using approximation methods, with the knowledge of the server's model and one sample's posteriors. Empirical evidence has shown that the reconstructed intermediate results are close to the original ones and can be used to infer samples' raw features and relations (He, Zhang, and Lee 2019). These leakages of information reveal that even intermediate results carry too much information.

Adversarial Attacks As VFL is a distributed system, the presence of malicious parties is inevitable. This presents a challenge for applications of VFL, as it is necessary to mitigate adversarial attacks when one party uploads crafted adversarial features. Existing defense methods are available but can result in additional costs. A key question is whether there is a more economical way to achieve the same goal or reduce the overall cost of mitigating adversarial attacks in VFL. From the empirical study on the adversarial attacks, we find that FP numbers may exacerbate the sensitivity of the deep learning model, thereby making it more vulnerable to adversarial attacks.

Based on the analysis above, we speculate that the usage of FP numbers may be a common factor contributing to the issues encountered in VFL. To address these issues, we propose a new architecture called the rounding layer, which converts intermediate results to integers. This architecture can be easily incorporated into existing VFL frameworks. The benefits of this approach include:

1. Enabling HE calculations without transformation;
2. Compressing the intermediate results;
3. Expanding the robust boundary of samples.

The proposed architecture addresses the following questions:

- **How to continue backward propagation through the rounding layer?**

[*] Corresponding authors.

To solve this problem, we adopt a Straight Through Estimator (Bengio, Léonard, and Courville 2013) that estimates the gradient of the rounding layer. Specifically, the gradients passing through the rounding layer are set identically in the backward propagation. This operation enables the continuation of backward propagation, and we provide an analysis in the following section.

- **The trade-off of rounding the intermediate results and the task's performance.**

  To address this concern, we provide an error bound for the rounding layer and empirically demonstrate the performance of the classification task on three popular datasets. In addition to compressing information, we analyze the effect of the rounding operation on privacy protection from the perspective of differential privacy (DP) (Dwork and Roth 2014).

- **How to quantify the gains of the robust boundary?**

  To address this problem, we provide two analysis: First, we evaluate the defense effectiveness against a Projected Gradient Descent (PGD) attack (Madry et al. 2017). Second, we calculate the statistical radius of samples from different implementations using randomized smoothing (Cohen, Rosenfeld, and Kolter 2019).

Our analysis and empirical results demonstrate that the combination of the rounding layer in VFL can effectively save computational and communication resources, compress information, provide privacy protection, improve robustness, and most importantly, preserve the main task's performance. The contributions of the paper are as follows:

- We investigate the existing open problems in VFL, and identify one fundamental factor, FP numbers, that causes the issues.
- We propose a new architecture, the rounding layer, that supports converting intermediate results to integers in VFL and demonstrate how to implement it.
- We analyze the effectiveness of the rounding layer in reducing computational and memory overhead, as well as its error bound and level of privacy protection.
- We empirically demonstrate the effectiveness of the rounding operation in preserving consistency on feature attribution and mitigating adversarial attacks.

Background

Vertical Federated Learning

Vertical federated learning (Liu et al. 2022; Yang et al. 2019) is a trending solution for collaborative learning. It enables each client to train a deep learning network, known as a local model, on its local data first. The extracted intermediate results are then sent to the server for aggregation and further computation, such as calculating posteriors for classification tasks. The updates for intermediate results are computed and sent back to the corresponding clients, and this process continues until the performance converges. In this way, raw data is kept locally, and only intermediate results and corresponding updates are exchanged, providing privacy protection.

VFL allows each client the freedom to choose their local models, making it flexible in application to different data modalities. However, due to communication costs and the size of the overlapped user space, current use cases mainly involve two-party collaboration.

Homomorphic Encryption Homomorphic encryption (Gentry 2009) is a popular technique used in Secure Multiparty Computation that is also employed in VFL to ensure the privacy of intermediate results. It enables aggregation and computation to be encrypted.

Two streams of HE exist: Fully Homomorphic Encryption (FHE) and Partially Homomorphic Encryption (PHE). FHE supports both addition and multiplication after encryption, while PHE supports only one of them. Although FHE is more secure, PHE has an advantage in efficiency due to its simpler mechanism. Therefore, in VFL, we choose Paillier Homomorphic Encryption (Paillier 1999) as the implementation method, which is a popular implementation of PHE that supports addition.

Since the foundation of Homomorphic Encryption is number theory, it cannot be directly applied to real numbers. A practical way to address this issue is to choose an appropriate precision and then transform real numbers to fixed-point numbers for further processing.

Differential Privacy

Differential privacy is a widely used method in protecting data privacy. Formally, it can be defined as follows.

Definition 1 *A mechanism $\mathcal{M}$ is considered to be (ϵ, δ)-differential privacy if, for all adjacent datasets $\mathcal{D}$ and $\mathcal{D}'$, and for all possible subsets of results $\mathcal{S}$, the following holds:*

$$\mathbb{P}[\mathcal{M}(\mathcal{D}) \in \mathcal{S}] \leq e^{\epsilon} * \mathbb{P}[\mathcal{M}(\mathcal{D}') \in \mathcal{S}] + \delta,$$

where $\mathbb{P}$ denotes the probability, ϵ denotes the privacy budget, and δ denotes the probability of failure.

In practice, the value of ϵ and δ should be as small as possible to provide a strong protection of privacy. However, the smaller the values, the larger the noise added to the data by $\mathcal{M}$, causing the degradation of the utility.

Adversarial Attack

Adversarial attack is one of the most concerning problems in the field of machine learning security in recent years (Szegedy et al. 2013; Madry et al. 2017; Goodfellow, Shlens, and Szegedy 2015). To perform the adversarial attack, an adversary needs to craft an perturbation for a target sample, which aims to fool a specific deep learning model during the inference phase. Formally, the objective function can be formulated as follows:

$$\min_{\mathbf{x}' \in \mathcal{B}_{\gamma}(\mathbf{x})} \mathcal{L}(\mathbf{x}', y_t; \theta), \tag{1}$$

where $\mathcal{L}$ measures the loss between the posteriors of the perturbed sample, $\mathbf{x}'$, and the target class y_t; θ is the parameter of f, and $\mathcal{B}_{\gamma}(\mathbf{x}) = \{\mathbf{x}' | ||\mathbf{x}' - \mathbf{x}||_p \leq \gamma\}$ depicts a norm ball at $\mathbf{x}$, where $||\cdot||_p$ denotes the L_p norm distance and γ denotes the radius. The term of $\mathcal{B}_{\gamma}(\mathbf{x})$ is to ensure that the perturbation is stealthy to avoid human check.

Fast Gradient Sign Method (FGSM) (Goodfellow, Shlens, and Szegedy 2015) proposed an efficient way to solve the above equation by optimizing the perturbation as:

$$\mathbf{x}' = \mathbf{x} - \eta * sign(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y_t; \theta)), \tag{2}$$

where η denotes the step size of the optimization, $sign(\cdot)$ returns the sign of the values, and ∇ denotes the gradients. However, with the accumulation of the optimization, $\mathbf{x}'$ may finally exceed the norm ball. PGD proposed a clipping operation at each step to keep the restriction.

Vertical Federated Learning with Rounding

Rounding Layer Design

We propose a new architecture, called the rounding layer, to implement the rounding operation in VFL. There are two key steps in the design of the rounding layer.

Forward According to IEEE Standard for Floating-Point Arithmetic (IEEE 2019), there are five rounding rules. The first two rules round to a nearest value, while the others are called directed roundings:

- **Rounding to Nearest:**
  1. Ties to even rounding: If the number falls midway, it is rounded to the nearest value with an even least significant digit.
  2. Ties away from zero (or ties to away) rounding: If the number falls midway, it is rounded to the nearest value above (for positive numbers) or below (for negative numbers).
- **Directed Rounding:**
  1. Towards 0 rounding: directed rounding towards zero (also known as truncation).
  2. Towards $+\infty$ rounding: directed rounding towards positive infinity (also known as ceiling).
  3. Towards $-\infty$ rounding: directed rounding towards negative infinity (also known as floor).

Based on the precision of the available rounding methods, we choose rounding to nearest as our rounding operation. We implement it with $ceil(\cdot)$ function, also denoted by $\lceil \cdot \rceil$, in PyTorch (Paszke et al. 2019), which can be formulated as follows:

$$round(x) = \lceil x - 0.5 \rceil. \tag{3}$$

In the following, we use $[\cdot]$ to represent the round operation, $round(\cdot)$, for simplicity.

Backward Using the $[\cdot]$ creates the problem of gradient vanishing. To address this issue, we introduce the STE to estimate the gradients. Let $\mathcal{L}(\cdot)$ denote the loss function, the gradient of $\mathbf{x}$ is estimated by STE as follows:

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mathbf{x}} \simeq \frac{\partial \mathcal{L}(\mathbf{x})}{\partial [\mathbf{x}]}. \tag{4}$$

Equation (4) means that, during the backward process, the gradients of the corresponding intermediate results will be set exactly the same as the gradients calculated on their conversions. This enables the training of deep learning models in VFL to continue.

Algorithm 1: Rounding in Vertical Federated Learning

```
Require: clients' bottom models $\{f_i\}_{i=1}^N$, server's top model $f_{top}$.
Ensure: trained $\{f_i\}_{i=1}^N$, $f_{top}$ for inference.
for each epoch do
    for each batch $(\mathbf{X}, \mathbf{Y})$ do
        During forward process:
        for At each $Client_i$ do
            $\mathbf{Emb}_i \leftarrow f_i(\mathbf{X}_i)$
            $\mathbf{V}_i \leftarrow [\mathbf{Emb}_i]$
            Send $\mathbf{V}_i$ to the server
        end for
        At the server:
        $\mathbf{V} \leftarrow concate(\{\mathbf{V}_i\}_{i=1}^N)$
        $\mathcal{L} \leftarrow cross\_entropy(f_{top}(\mathbf{V}), \mathbf{Y})$
        During backward process:
        At the server:
        for each $\mathbf{V}_i$ do
            calculate $\frac{\partial \mathcal{L}}{\partial \mathbf{V}_i}$
            send $\frac{\partial \mathcal{L}}{\partial \mathbf{V}_i}$ to the corresponding client
        end for
        for At each $Client_i$ do
            $\frac{\partial \mathcal{L}}{\partial \mathbf{Emb}_i} \leftarrow \frac{\partial \mathcal{L}}{\partial \mathbf{V}_i}$
            update the following parameters of $f_i$
        end for
    end for
end for
```

Algorithm 1 summarizes the process. During each epoch of training, N clients first extract intermediate results, $\mathbf{Emb}_i$, from their bottom model, f_i, of raw data $\mathbf{X}_i$ in the forward process. Then, the rounding layer converts $\mathbf{Emb}_i$ to integers by $[\cdot]$. The conversions are sent to the server, who aggregates these conversions and uses f_{top} to do the final computation. During backward process, the server calculates each $\mathbf{V}_i$'s gradients and send them back to corresponding clients. Using STE, each client passes the received gradients through the rounding layer without modification and updates the parameters of their bottom models.

Computation and Memory Efficiency

Since the intermediate results are represented as integers, there is no need of extra conversion cost from the FP type to an Int type in order to employ Paillier Homomorphic Encryption. Therefore, comparing to the solution of conversion to fixed-point numbers, the rounding operation does not increase the time cost. In addition, according to the findings (Abdel-Aziz et al. 2021), the Int type can further accelerate computation, which helps improve training efficiency.

In PyTorch's default setting, 32 bits are used to store a floating-point type tensor, while only 8 bits are used for an integer type tensor. As a result, the memory compression theoretically achieves a 4× reduction, which also saves considerable band cost during the communication phase.

Error Bound

Although we have solved the challenge of gradient vanishing caused by the rounding operation, it is still a question whether such a design can promise a close performance. This section formally explores the error bound for local optima after rounding. Specifically, we present the following theorem for the error bound.

Theorem 1 *Given $\boldsymbol{x} = \boldsymbol{z} + \boldsymbol{r}$, where $\boldsymbol{z} \in \mathbb{Z}^d$, and $\boldsymbol{r} \in [-\frac{1}{2}, \frac{1}{2}]^d$. Assume that for a specific class, the top model's prediction can be approximated by a 2-times differential function $g : \mathbb{R}^d \to \mathbb{R}$. Then, let $\Delta = g(\boldsymbol{x}) - g(\boldsymbol{z})$, we have:*

$$||\Delta||_2 \leq \sum_{||\boldsymbol{\alpha}||_1=1} \frac{1}{2^{\boldsymbol{\alpha}}} || \frac{D^{\boldsymbol{\alpha}} g(\boldsymbol{z})}{\boldsymbol{\alpha}!} ||_2 + \sum_{||\boldsymbol{\beta}||_1=2} \frac{1}{2^{\boldsymbol{\beta}}} \dot{R}_{\boldsymbol{\beta}}(\boldsymbol{z}),$$

where $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}^d$ are multi-index notation, $||\boldsymbol{\alpha}||_1 = \alpha_1 + \cdots + \alpha_d$, and $\boldsymbol{\alpha}! = \alpha_1! \cdots \alpha_d!$; $D^{\boldsymbol{\alpha}} g = \frac{\partial^{||\boldsymbol{\alpha}||_1} g}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}$; $\dot{R}_{\boldsymbol{\beta}}(\boldsymbol{z}) = \frac{1}{\boldsymbol{\beta}!} \max_{||\boldsymbol{\alpha}||_1 = ||\boldsymbol{\beta}||_1} \max_{\boldsymbol{y} \in \mathcal{B}_{\frac{1}{2}}(\boldsymbol{z})} ||D^{\boldsymbol{\alpha}} g(\boldsymbol{y})||$, and $\mathcal{B}_{\frac{1}{2}}(\boldsymbol{z})$ denotes the norm ball of $\boldsymbol{z}$ with the radius of $\frac{1}{2}$.

Proof With the notation and assumption of Theorem 1, we can reformulate $g(\mathbf{x})$ according to the Multivariate Version of Taylor's Theorem (Wheeden and Zygmund 1977) as follows:

$$\begin{aligned} g(\mathbf{x}) &= \sum_{||\boldsymbol{\alpha}||_1 \leq 1} \frac{D^{\boldsymbol{\alpha}} g(\mathbf{z})}{\boldsymbol{\alpha}!} \cdot \mathbf{r}^{\boldsymbol{\alpha}} + \sum_{||\boldsymbol{\beta}||_1=2} R_{\boldsymbol{\beta}}(\mathbf{x}) \cdot \mathbf{r}^{\boldsymbol{\beta}}, \\ R_{\boldsymbol{\beta}}(\mathbf{x}) &= \frac{||\boldsymbol{\beta}||_1}{\boldsymbol{\beta}!} \int_0^1 (1-t)^{||\boldsymbol{\beta}||_1 - 1} D^{\boldsymbol{\beta}} g(\mathbf{z} + t(\mathbf{x} - \mathbf{z})) dt, \end{aligned} \tag{5}$$

where t is an auxiliary variable. Due to the continuity of second order partial derivatives, we can obtain the uniform estimates:

$$||R_{\boldsymbol{\beta}}(\mathbf{x})||_2 \leq \frac{1}{\boldsymbol{\beta}!} \max_{||\boldsymbol{\alpha}||_1 = ||\boldsymbol{\beta}||_1} \max_{\mathbf{y} \in \mathcal{B}_{\frac{1}{2}}(\mathbf{z})} ||D^{\boldsymbol{\alpha}} g(\mathbf{y})||. \tag{6}$$

Let $\Delta = g(\mathbf{x}) - g(\mathbf{z})$, we have the error bound after rounding as follows:

$$\begin{aligned} ||\Delta||_2 &= || \sum_{||\boldsymbol{\alpha}||_1=1} \frac{D^{\boldsymbol{\alpha}} g(\mathbf{z})}{\boldsymbol{\alpha}!} \cdot \mathbf{r}^{\boldsymbol{\alpha}} + \sum_{||\boldsymbol{\beta}||_1=2} R_{\boldsymbol{\beta}}(\mathbf{x}) \cdot \mathbf{r}^{\boldsymbol{\beta}} ||_2 \\ &\leq \sum_{||\boldsymbol{\alpha}||_1=1} || \frac{D^{\boldsymbol{\alpha}} g(\mathbf{z})}{\boldsymbol{\alpha}!} ||_2 \cdot ||\mathbf{r}^{\boldsymbol{\alpha}}||_2 + \sum_{||\boldsymbol{\beta}||_1=2} ||R_{\boldsymbol{\beta}}(\mathbf{x})||_2 \cdot ||\mathbf{r}^{\boldsymbol{\beta}}||_2 \\ &\leq \sum_{||\boldsymbol{\alpha}||_1=1} \frac{1}{2^{\boldsymbol{\alpha}}} || \frac{D^{\boldsymbol{\alpha}} g(\mathbf{z})}{\boldsymbol{\alpha}!} ||_2 + \sum_{||\boldsymbol{\beta}||_1=2} \frac{1}{2^{\boldsymbol{\beta}}} ||R_{\boldsymbol{\beta}}(\mathbf{x})||_2 \\ &\leq \sum_{||\boldsymbol{\alpha}||_1=1} \frac{1}{2^{\boldsymbol{\alpha}}} || \frac{D^{\boldsymbol{\alpha}} g(\mathbf{z})}{\boldsymbol{\alpha}!} ||_2 + \sum_{||\boldsymbol{\beta}||_1=2} \frac{1}{2^{\boldsymbol{\beta}}} \dot{R}_{\boldsymbol{\beta}}(\mathbf{z}). \end{aligned} \tag{7}$$

where $\dot{R}_{\boldsymbol{\beta}}(\mathbf{z}) = \frac{1}{\boldsymbol{\beta}!} \max_{||\boldsymbol{\alpha}||_1 = ||\boldsymbol{\beta}||_1} \max_{\mathbf{y} \in \mathcal{B}_{\frac{1}{2}}(\mathbf{z})} ||D^{\boldsymbol{\alpha}} g(\mathbf{y})||$.

Note that the above derivation is a theoretical guarantee. In the following section, we further illustrate the precision preserved by the rounding layer through evaluation on the performance of the classification tasks.

Privacy Measurement

This section measures the privacy protection provided by the rounding layer. As a popular technique, DP can provide a formal privacy guarantee in terms of privacy budget ϵ and failure probability δ. A common way to implement (ϵ, δ)-DP is adding Laplace noises to the output of a function f, such as:

$$\mathcal{M}(\mathbf{x}) = f(\mathbf{x}) + Lap(\frac{s}{\epsilon}), \tag{8}$$

where s denotes the sensitivity of $ceil(\cdot)$, and $Lap(\frac{s}{\epsilon})$ denotes sampling from Laplace distribution with center 0 and scale $\frac{s}{\epsilon}$.

Following the derivation in (Pham et al. 2022), we can also add a perturbation to the rounded embeddings for privacy analysis. Let $\mathcal{M}_r$ denote the mechanism of the rounding layer and $\mathcal{M}$ denote the mechanism of adding Laplace noises. Then, we can formulate $\mathcal{M}_r$ as follows:

$$\mathcal{M}_r(\mathbf{x}) = [\mathcal{M}(\mathbf{x})] = [[\mathbf{x}] + Lap(\frac{1}{\epsilon})], \tag{9}$$

where the sensitivity of $f(\mathbf{x}) = [\mathbf{x}]$ is 1. The first equation is because, from the server's perspective, it always receives the embeddings in integer format. The second equation follows the analysis of DP.

If $||Lap(\frac{1}{\epsilon})||_2 < \frac{1}{2}$, then $\mathcal{M}_r(\mathbf{x}) = [\mathbf{x}]$. It means that the rounding operation naturally tolerates a small latent noise. Let $cdf(\cdot)$ denote the cumulative distribution function of Laplace, we have:

$$\begin{aligned} \mathbb{P}[|Lap(\frac{1}{\epsilon})| < \frac{1}{2}] &= [cdf(\frac{1}{2}) - cdf(-\frac{1}{2})] \\ &= 1 - exp(-\frac{\epsilon}{2}). \end{aligned} \tag{10}$$

Regarding the Definition 1, the rounding operation can provide ϵ-DP with probability $1 - exp(-\frac{\epsilon}{2})$. This means the rounding operation naturally provides (ϵ, δ)-DP with $\delta = exp(-\frac{\epsilon}{2})$, and the lower bound of ϵ is $-2log(\delta)$. Therefore, the rounding operation could provide better privacy compared to the implementation without conversion.

Experiments and Results

Experimental Setup

Datasets We involve the following datasets in the study:

- **MNIST** (Lecun et al. 1998) is a widely used benchmark, which are handwritten digits with a training set of 60,000 examples, and a test set of 10,000 examples.
- **Fashion-MNIST** (Xiao, Rasul, and Vollgraf 2017) is a newly proposed dataset, which contains images of different types of clothes, with a training set of 60,000 examples, and a test set of 10,000 examples.
- **CIFAR10** (Krizhevsky, Hinton et al. 2009) is also a well-known image dataset, which consists of 60,000 colour images.

The two-party VFL is considered as the primary case in evaluation, as it is a more common case than the multi-party scenario in practice. To simulate VFL scenario, if the two parties have the same number of features, the image will be split in the middle (Fu et al. 2022).

Models There are two parts of models in VFL as follows:

- **Bottom Model** uses pretrained ResNet (He et al. 2016) to extract intermediate results. We modify the input and the output layers to suit the VFL training, including different channels of the images and the dimensional size of the intermediate results. The default size is set at 16.
- **Top Model** is a customized Multilayer Perceptron (MLP) (Ruppert 2004). Each layer is a composition of one linear layer and one activation layer, i.e., ReLU (Agarap 2018). By default the depth of the top model is set as 3.

The training details are set as follows: training epochs of 30, batch size of 256, learning rate of 10^{-3}, and weight decay of 5×10^{-4}. Adam (Kingma and Ba 2017) is the optimizer used by default.

Baselines B-SL (Pham et al. 2022) and HashVFL (Qiu et al. 2022b) are two related works that utilized deep hashing to binarize the intermediate results in VFL. Both studies demonstrated that binarization can lead to acceleration and privacy improvements.

However, the use of deep hashing can result in significant information loss, raising questions about the accuracy preservation and interpretability of their methods compared to ours. The former question is related to the original motivation behind VFL, while the latter is concerned with the rewards of providing features in VFL.

Therefore, we choose the binarizing operation as one baseline, which is denoted by 'Binary'. Additionally, we use the original framework that without extra modification as the baseline, denoted by 'Base'.

Performance of Classification Tasks

We conduct our experiments on two distinct parameters, namely dimensional size and feature ratio. The dimensional size refers to the size of intermediate results in VFL, while the feature ratio denotes the proportion of features contributed by each client, relative to the total number of features. When it comes to image data, we use the number of columns to indicate the number of features.

Dimensional Size Analysis The accuracy results obtained on different datasets with varying dimensional sizes are summarized in Table 1. We have transformed the dimensional size by a factor of 2, ranging from 8 to 128. The feature ratio is fixed at 50% for all clients.

Our results indicate that the rounding layer does not lead to significant accuracy loss in classification tasks. In some cases, our proposed architecture even outperforms the base architecture. This could be due to redundancy in FP numbers, which carry more information than clustering, causing confusion in classification.

However, the binary transformation results in a reduction in accuracy, particularly on the CIFAR10 dataset, where the accuracy drops from 75.34% to 70.07% when the dimensional size is 16. This suggests that our proposed transformation better preserves the performance of the main task.

Furthermore, our results show that the dimensional size does not appear to substantially affect the performance. We speculate that this is because the bottom models, ResNet, are proficient at extracting valid information. This finding points to an interesting application scenario for VFL, which involves combining large models with multiple modalities for joint modeling.

Dataset	Arch.	Dimensional Size				
		d=8	d=16	d=32	d=64	d=128
MNIST	Base	98.41	**98.77**	98.33	98.50	**98.70**
	Binary	97.66	98.57	98.28	97.91	98.34
	Round	**98.43**	98.66	**98.39**	**98.66**	98.31
Fashion	Base	**90.88**	90.87	89.52	**90.75**	90.29
	Binary	90.52	89.69	90.30	89.44	89.30
	Round	90.84	**90.91**	**90.70**	90.66	**90.52**
CIFAR10	Base	**74.59**	75.34	**75.76**	75.15	75.04
	Binary	70.13	70.07	69.41	71.87	70.06
	Round	73.41	**75.67**	74.74	**75.42**	**75.33**

Table 1: Comparison with different dimensional sizes.

Dataset	Arch.	Feature Ratio				
		r=0.1	r=0.2	r=0.3	r=0.4	r=0.5
MNIST	Base	**99.02**	**99.13**	98.96	**98.77**	**98.77**
	Binary	98.85	99.11	98.93	98.61	98.57
	Round	98.83	99.06	**99.13**	98.62	98.66
Fashion	Base	**91.85**	91.51	91.37	90.88	90.87
	Binary	91.35	91.17	91.06	90.95	89.69
	Round	91.67	**91.78**	**91.59**	**91.83**	**90.91**
CIFAR10	Base	**81.79**	**79.82**	76.79	**75.27**	75.34
	Binary	79.32	78.51	75.39	74.01	70.07
	Round	80.92	78.82	**77.19**	74.97	**75.67**

Table 2: Comparison with different feature ratios.

Feature Ratio Analysis The impact of feature ratio on classification task performance is summarized in Table 2. We vary the feature ratio of one party from 10% to 50% to observe the differences.

Our results show that the proposed architecture effectively preserves the accuracy of the main task. However, the binary operation loses a considerable amount of information, notably on the CIFAR10 dataset, where the accuracy decreases by 2.47% with a feature ratio of 10%, and 5.27% with a feature ratio of 50%.

Another interesting finding is that when one party has a higher proportion of complete features, i.e., a feature ratio of 10% in this case, the accuracy is higher than when each party has less complete features, i.e., a feature ratio of 50%. This is plausible as incomplete features, such as a split dog face, may have weak signals in extracted intermediate results, with the background having a more significant impact.

Feature Attribution

An essential aspect of applying VFL is to interpret the importance of features (Wang 2019). This analysis relates to the interests of each party, particularly the value assessment of the provided features.

It is expected that binarization will inevitably affect the distribution of importance, while rounding should have a lesser impact on this distribution. In the following, we verify this speculation through experiments.

Attribution Methods Before the experiments, we introduce three popular attribution methods involved as follows:

- *Integrated Gradients.* Integrated Gradients (Sundararajan, Taly, and Yan 2017) represents the integral of gradients with respect to inputs along the path from a given baseline to input. It can be formulated as follows:
$$IG_i(x) := (x_i - x'_i) \times \int_{t=0}^{1} \frac{\partial f(x' + t(x - x'))}{\partial x_i} dt, \tag{11}$$
where i denotes the i-th dimension of input x. The integral can be approximated by Riemann Sum or Gauss Legendre quadrature rule.
- *DeepLIFT.* DeepLIFT (Shrikumar, Greenside, and Kundaje 2017) is also a back-propagation based approach that attributes a change to inputs based on the differences between the inputs and corresponding baselines. Specifically, DeepLIFT uses the concept of multipliers to measure specific neurons for the difference in outputs. Formally, it can be described as follows:
$$m_{\Delta \mathbf{x} \Delta n} := \frac{C_{\Delta \mathbf{x} \Delta n}}{\Delta \mathbf{x}}, \tag{12}$$
where $\Delta\mathbf{x}$ denotes the difference between the input $\mathbf{x}$ and the reference, Δn represents the difference between the target neuron n and the reference, and C accumulates the contribution of $\Delta\mathbf{x}$ to Δn.
- *Feature Ablation.* Feature Ablation (Kokhlikyan et al. 2020) is a perturbation based approach to compute attribution, involving replacing each input feature with a given reference value (e.g., 0), and computing the difference in output. For example, for images, one can group an entire segment or region and ablate it, measuring the importance of the segment.

We use the open source library, Captum (Kokhlikyan et al. 2020), to implement the aforementioned methods, which offers excellent toolkits for explaining deep learning models.

Evaluation Metrics In VFL, a practical and equitable method to assess each party's contribution is to measure the importance of their submitted intermediate results, presented as an array that represents the significance of each dimension. We propose the following metrics for evaluation:

- *Euclidean Distance*, which is used to compute the numerical absolute error of the baseline's interpretations and processed results.
- *Correlation Distance*, which is used to measure the similarity between the baseline's interpretations and others.
- *Kendall's τ* (Kendall 1938), which is used to measure the rank correspondence between two interpretations, since preserving ranking is also important.

We use the open source library, SciPy (Virtanen et al. 2020), to implement the computation of different metrics.

Consistency Evaluation To compare the methods, we keep the dimensional size at 16, feature ratio at 50%, and target class at 0 (e.g., digit '0' in MNIST). The results are shown in Table 3.

Our results indicate that the rounding architecture preserves consistency better than the binary design for all three methods. For example, on the CIFAR10, our design achieves an absolute error of 0.17, correlation distance of 0.19, and Kendall's τ of 0.47 with a p-Value of 0.0001, while the binary design obtains an Euclidean distance of 0.34, correlation distance of 1.09, and Kendall's τ of -0.13 with a p-Value of 0.31. Note that for the first two metrics, lower values are better, while for Kendall's τ, higher values are better, and small p-Values are expected.

Our comparison of feature attribution consistency demonstrates that our architecture has a small absolute error and close correlation with the original one, and the ranking of importance for different dimensions can be maintained.

Mitigating Adversarial Attack

Threat Model In our evaluation, we assume the strongest possible adversary who possesses complete knowledge of the submitted intermediate results and the parameters of the top model. Thus, the adversary can conduct a white-box attack. The adversary's objective is to change the prediction of a target sample to the desired class, which is fixed as class 0 in the following experiments.

We use PGD attack in evaluation, which can be described as follows:

$$\begin{cases} \mathbf{x}_m = \mathbf{x}_{m-1} - \eta * sign(\nabla_{\mathbf{x}_{m-1}} \mathcal{L}(\mathbf{x}_{m-1}, y_t; \theta)) \\ \boldsymbol{\phi}_m = clip(\mathbf{x}_m - \mathbf{x}_0, \omega) \\ \mathbf{x}_m = \mathbf{x}_0 + \boldsymbol{\phi}_m, \end{cases} \tag{13}$$

where $\mathbf{x}_0$ denotes the original intermediate results, $\mathbf{x}_m$ denotes the perturbed adversarial results at m-th optimization, and $clip(\mathbf{x}_m - \mathbf{x}_0, \omega)$ denotes the restriction that clips the perturbation $\boldsymbol{\phi}_m$ to a given threshold, which is $(-\omega, \omega)$.

Attack Success Rate Reduction We evaluate the security of different architectures under varying combinations of threshold ω and step size s. The evaluation comprises two aspects: the preserved accuracy and the attack success rate with perturbation. The difference between these two aspects is that the perturbed intermediate results may be misclassified but not predicted to the desired class (sometimes referred to as untargeted and targeted attacks in literature). We set the threshold as 1 and 2, respectively, with the step size set at 0.1 and 1.0. We randomly select 100 samples for evaluation. Table 4 summarizes the results.

Note that during the adversarial attack, the perturbed intermediate results must still remain in the binary or integer form. Therefore, when the step size is 0.1, the perturbation cannot change the value of the intermediate results. The results in Table 4 confirm this point, where the preserved accuracy remains the same, and the attack success rate is 0 with binary and rounding. However, if the step size is increased to 1.0, all architectures struggle to thoroughly mitigate the adversarial attack, particularly when the threshold is 2. Nevertheless, our rounding operation is more robust in preserving accuracy and defending against attacks, as shown in Table 4.

An interesting phenomenon is that as the step size expands, the attack success rate drops in the base architecture.

Dataset	Arch.	Methods											
		Integrated Gradients				DeepLIFT				Feature Ablation			
		Euc.	Cor.	Kendall's τ		Euc.	Cor.	Kendall's τ		Euc.	Cor.	Kendall's τ	
				Stats	p-Value			Stats	p-Value			Stats	p-Value
MNIST	Binary	0.2646	0.8939	0.0282	0.8344	0.2853	0.8069	0.0887	0.4887	0.2863	0.7793	0.1210	0.3415
	Round	**0.1048**	**0.0849**	**0.5726**	0.0001	**0.1860**	**0.3191**	**0.3710**	0.0025	**0.1886**	**0.2992**	**0.3750**	0.0022
Fashion	Binary	0.2952	0.9072	0.0968	0.4491	0.2853	0.8069	0.0887	0.4887	0.2863	0.7793	0.1210	0.3415
	Round	**0.1552**	**0.1925**	**0.5847**	0.0001	**0.1860**	**0.3191**	**0.3710**	0.0025	**0.1886**	**0.2992**	**0.3750**	0.0022
CIFAR10	Binary	0.3394	1.0851	-0.1290	0.3096	0.2952	0.9408	-0.0605	0.6408	0.3265	0.9771	-0.0847	0.5092
	Round	**0.1702**	**0.1942**	**0.4718**	0.0001	**0.1644**	**0.2926**	**0.3427**	0.0055	**0.1688**	**0.2503**	**0.3790**	0.0020

Table 3: Evaluation results for feature attribution consistency. 'Euc.' represents the Euclidean distance, while 'Cor.' represents the Correlation distance. Smaller distances indicate better results. For Kendall's τ, higher stats indicate better performance.

We speculate that this is because a large step size also results in a loss of precision in generating optimum perturbations.

Overall, our rounding operation demonstrates stronger ability to mitigate adversarial attacks than the baselines and the binary architecture. To quantify the improvement, we further present the analysis of the certified robust radius in the following section.

Certified Robust Radius There are various methods available for quantifying the adversarial robustness of deep learning models. To account for generalization, we use randomized smoothing (Cohen, Rosenfeld, and Kolter 2019) to compute the certified robustness radius for samples, which is independent of any specific model.

Specifically, randomized smoothing method aims to construct a "smoothed" classifier g from the base classifier $f : \mathbb{R}^d \to \mathcal{Y}$. When queried at $\mathbf{x}$, g returns whichever class f is most likely to return when $\mathbf{x}$ is perturbed by isotropic Gaussian noise:

$$g(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \mathbb{P}(f(\mathbf{x} + \boldsymbol{\xi}) = y), \tag{14}$$

where $\boldsymbol{\xi} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. Since it is not possible to exactly evaluate the prediction of g around $\mathbf{x}$, Monte Carlo algorithms is used for assessment.

Theorem 2 *Let $f : \mathbb{R}^d \to \mathcal{Y}$ be any deterministic or random function, and let $\boldsymbol{\xi} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. Let g be defined as the smoothed classifier. Suppose $y_A \in \mathcal{Y}$ and $\underline{p_A}$, $\overline{p_B} \in [0, 1]$ satisfy:*

$$\mathbb{P}(f(\boldsymbol{x} + \boldsymbol{\xi}) = y_A) \geq \underline{p_A} \geq \overline{p_B} \geq \max_{y \neq y_A} \mathbb{P}(f(\boldsymbol{x} + \boldsymbol{\xi}) = y).$$

Then, $g(\boldsymbol{x} + \boldsymbol{\xi}) = y_A$ for all $||\boldsymbol{\xi}||_2 < \mathcal{R}$, where

$$\mathcal{R} = \frac{\sigma}{2}(\Phi^{-1}(\underline{p_A}) - \Phi^{-1}(\overline{p_B})).$$

According to Theorem 2 (Cohen, Rosenfeld, and Kolter 2019), we can get that g is robust around $\mathbf{x}$ within the ℓ_2 radius $\mathcal{R} = \frac{\sigma}{2}(\Phi^{-1}(p_A) - \Phi^{-1}(p_B))$, where Φ^{-1} is the inverse of the standard Gaussian CDF, p_A denotes the probability of the most probable class y_A is returned, and p_B denotes the other classes. The result also holds if we replace p_A with a lower bound $\underline{p_A}$ and p_B with an upper bound $\overline{p_B}$.

We randomly select 1,000 samples and conduct 10^6 samples for each to assess the radius. The variance σ is fixed at 1. Table 5 summarizes the certified robustness radius results for different architectures. Experimental results demonstrate that the rounding operation enlarges the radius of robustness around each $\mathbf{x}$.

Additionally, we observe that if the perturbations exceed the restriction, binarization becomes weaker in mitigating adversarial attacks than the base architecture. This phenomenon is consistent with the results presented in Table 4. This may be the information loss caused by binarization results in the value of each bit being vital to the prediction, and any changes can significantly affect the final outcome.

Discussion

Combination of Integerized Gradients This paper addresses the challenges encountered during the forward process in VFL. However, there are still risks of leakage during the backward process. For example, Zhu et al. (Zhu, Liu, and Han 2019) revealed that the gradients could be utilized for reconstruction attacks. Concerning the efficacy of the rounding operation and the feasibility of gradient discretization (Dryden et al. 2016) and sparsification (Aji and Heafield 2017), we speculate that integerization could be a promising approach to address this issue.

Specialized Integer Tensor Operations Although half FP and binary values have been proven efficient with specialized operation designs, we believe that specialized operations for integer tensors could also accelerate training and inference in VFL. If the combination of Integerized Gradients is proven to be effective, this attempt will be valuable.

Conclusion

This paper proposes a novel architecture to tackle the challenges in VFL, including computational overhead, privacy protection, and security concerns arising from adversarial attacks. We have theoretically analyzed the advantages of the rounding layer in terms of computation efficiency and memory reduction, rounding error bounds, and privacy protection from a DP perspective. Empirical studies indicate that the proposed rounding layer can preserve the model's performance, maintain consistency with the interpretation of the original framework, and mitigate adversarial attacks.

Acknowledgments

This work is supported by Ant Group through Ant Research Intern Program and also partly supported by the National

Dataset	Threshold ω	Step Size s	Accuracy			Preserved Accuracy			Attack Success Rate		
			Base	Binary	Round	Base	Binary	Round	Base	Binary	Round
MNIST	1	0.1	97%	**98%**	97%	15%	98%	**97%**	85%	0	**0**
		1.0	97%	**98%**	97%	14%	22%	**56%**	74%	64%	**43%**
	2	0.1	97%	**98%**	97%	0	98%	**97%**	100%	0	**0**
		1.0	97%	**98%**	97%	0	0	**0**	100%	100%	**96%**
Fashion	1	0.1	89%	83%	**92%**	73%	83%	**92%**	16%	0	**0**
		1.0	89%	83%	**92%**	77%	40%	**83%**	15%	42%	**10%**
	2	0.1	89%	83%	**92%**	12%	83%	**92%**	86%	0	**0**
		1.0	89%	83%	**92%**	12%	0	**14%**	**82%**	94%	83%
CIFAR10	1	0.1	**79%**	69%	77%	22%	69%	**77%**	76%	0	**0**
		1.0	**79%**	69%	77%	28%	9%	**59%**	66%	85%	**25%**
	2	0.1	**79%**	69%	77%	2%	69%	**77%**	98%	0	**0**
		1.0	**79%**	69%	77%	3%	0	**9%**	95%	64%	**43%**

Table 4: Attack success rate evaluation with different combinations of threshold and step size.

Dataset	Architecture					
	Base		Binary		Round	
	Mean	Std.	Mean	Std.	Mean	Std.
MNIST	2.20	0.70	2.15	0.64	**2.96**	1.15
Fashion	4.11	1.81	1.66	0.68	**4.28**	1.85
CIFAR10	1.92	1.70	1.12	0.81	**3.01**	2.17

Table 5: Certified robust radius.

Key Research and Development Program of China under No. 2022YFB3102100 and NSFC under No. 62172243 and 62102360.

References

Abdel-Aziz, H.; Shafiee, A.; Shin, J. H.; Pedram, A.; and Hassoun, J. 2021. Rethinking Floating Point Overheads for Mixed Precision DNN Accelerators. *ArXiv*, abs/2101.11748.

Agarap, A. F. 2018. Deep Learning using Rectified Linear Units (ReLU). *ArXiv*, abs/1803.08375.

Aji, A. F.; and Heafield, K. 2017. Sparse Communication for Distributed Gradient Descent. In *Conference on Empirical Methods in Natural Language Processing*.

Bengio, Y.; Léonard, N.; and Courville, A. C. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *ArXiv*, abs/1308.3432.

Cohen, J. M.; Rosenfeld, E.; and Kolter, J. Z. 2019. Certified Adversarial Robustness via Randomized Smoothing. In *International Conference on Machine Learning*.

Dryden, N.; Moon, T.; Jacobs, S. A.; and Essen, B. C. V. 2016. Communication Quantization for Data-Parallel Training of Deep Neural Networks. *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*, 1–8.

Dwork, C.; and Roth, A. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.*, 9: 211–407.

Fu, C.; Zhang, X.; Ji, S.; Chen, J.; Wu, J.; Guo, S.; Zhou, J.; Liu, A. X.; and Wang, T. 2022. Label Inference Attacks Against Vertical Federated Learning. In *USENIX Security Symposium*.

Gentry, C. 2009. Fully homomorphic encryption using ideal lattices. In *Symposium on the Theory of Computing*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. arXiv:1412.6572.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

He, Z.; Zhang, T.; and Lee, R. B. 2019. Model inversion attacks against collaborative inference. *Proceedings of the 35th Annual Computer Security Applications Conference*.

IEEE. 2019. IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, 1–84.

Kendall, M. G. 1938. A NEW MEASURE OF RANK CORRELATION. *Biometrika*, 30: 81–93.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Kokhlikyan, N.; Miglani, V.; Martin, M.; Wang, E.; Alsallakh, B.; Reynolds, J.; Melnikov, A.; Kliushkina, N.; Araya, C.; Yan, S.; and Reblitz-Richardson, O. 2020. Captum: A unified and generic model interpretability library for PyTorch. arXiv:2009.07896.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, Toronto, ON, Canada.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Liu, Y.; Fan, T.; Chen, T.; Xu, Q.; and Yang, Q. 2021. Fate: An industrial grade platform for collaborative learning with data protection. *The Journal of Machine Learning Research*, 22(1): 10320–10325.

Liu, Y.; Kang, Y.; Zou, T.; Pu, Y.; He, Y.; Ye, X.; Ouyang, Y.; Zhang, Y.; and Yang, Q. 2022. Vertical Federated Learning. *ArXiv*, abs/2211.12814.

Luo, X.; Wu, Y.; Xiao, X.; and Ooi, B. C. 2020. Feature Inference Attack on Model Predictions in Vertical Federated Learning. *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 181–192.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards Deep Learning Models Resistant to Adversarial Attacks. *ArXiv*, abs/1706.06083.

Moon, S.; and Lee, Y. 2020. An Efficient Encrypted Floating-Point Representation Using HEAAN and TFHE. *Secur. Commun. Networks*, 2020: 1250295:1–1250295:18.

Paillier, P. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *International Conference on the Theory and Application of Cryptographic Techniques*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc.

Pham, N. D.; Abuadbba, A.; Gao, Y.; Phan, K. T.; and Chilamkurti, N. K. 2022. Binarizing Split Learning for Data Privacy Enhancement and Computation Reduction. *IEEE Transactions on Information Forensics and Security*, 18: 3088–3100.

Qiu, P.; Zhang, X.; Ji, S.; Du, T.; Pu, Y.; Zhou, J.; and Wang, T. 2022a. Your Labels Are Selling You Out: Relation Leaks in Vertical Federated Learning. *IEEE Transactions on Dependable and Secure Computing*.

Qiu, P.; Zhang, X.; Ji, S.; Pu, Y.; and Wang, T. 2022b. All You Need Is Hashing: Defending Against Data Reconstruction Attack in Vertical Federated Learning. *ArXiv*, abs/2212.00325.

Ruppert, D. 2004. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. *Journal of the American Statistical Association*, 99: 567 – 567.

Shrikumar, A.; Greenside, P.; and Kundaje, A. 2017. Learning Important Features Through Propagating Activation Differences. In *International Conference on Machine Learning*.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic Attribution for Deep Networks. In *International Conference on Machine Learning*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I. J.; and Fergus, R. 2013. Intriguing properties of neural networks. *CoRR*, abs/1312.6199.

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272.

Wang, G. 2019. Interpret Federated Learning with Shapley Values. *ArXiv*, abs/1905.04519.

Wheeden, R. L.; and Zygmund, A. 1977. *Measure and integral : an introduction to real analysis*, chapter 1. CRC Press.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747.

Yang, Q.; Liu, Y.; Chen, T.; and Tong, Y. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2): 1–19.

Zhang, Q.; Wang, C.; Wu, H.; Xin, C.; and Phuong, T. V. 2018. GELU-Net: A Globally Encrypted, Locally Unencrypted Deep Neural Network for Privacy-Preserved Learning. In *IJCAI*, 3933–3939.

Zhang, Y.; and Zhu, H. 2020. Additively homomorphical encryption based deep neural network for asymmetrically collaborative machine learning. *arXiv preprint arXiv:2007.06849*.

Zhu, L.; Liu, Z.; and Han, S. 2019. Deep Leakage from Gradients. In *Neural Information Processing Systems*.