PraVFed: Practical Heterogeneous Vertical Federated Learning via Representation Learning

Shuo Wang, *Student Member, IEEE*, Keke Gai, *Senior Member, IEEE*, Jing Yu, *Member, IEEE*, Zijian Zhang, *Senior Member, IEEE*, and Liehuang Zhu, *Senior Member, IEEE*

***Abstract*—Vertical federated learning (VFL) provides a privacy-preserving method for machine learning, enabling collaborative training across multiple institutions with vertically distributed data. Existing VFL methods assume that participants passively gain local models of the same structure and communicate with active pary during each training batch. However, due to the heterogeneity of participating institutions, VFL with heterogeneous models for efficient communication is indispensable in real-life scenarios. To address this challenge, we propose a new VFL method called *Practical Heterogeneous Vertical Federated Learning via Representation Learning* (PraVFed) to support the training of parties with heterogeneous local models and reduce communication costs. Specifically, PraVFed employs weighted aggregation of local embedding values from the passive party to mitigate the influence of heterogeneous local model information on the global model. Furthermore, to safeguard the passive party's local sample features, we utilize blinding factors to protect its local embedding values. To reduce communication costs, the passive party performs multiple rounds of local pre-model training while preserving label privacy. We conducted a comprehensive theoretical analysis and extensive experimentation to demonstrate that PraVFed reduces communication overhead under heterogeneous models and outperforms other approaches. For example, when the target accuracy is set at 60% under the CINIC10 dataset, the communication cost of PraVFed is reduced by 70.57% compared to the baseline method. Our code is available at https://github.com/wangshuo105/PraVFed_main.**

***Index Terms*—Vertical federated learning, representation learning, heterogeneous model architecture, weight aggregation.**

I. Introduction

DEEP learning has garnered considerable attention in real-world applications. Traditional centralized deep-learning training methods require the aggregation of all clients' local data, allowing collectors direct access to this data, which can lead to serious privacy leakage for the clients. To address this issue, Federated Learning (FL) [1], [2], [3] is a novel paradigm of distributed machine learning designed to protect privacy. FL allows multiple participants to collaborate on learning while keeping data localized and confidential without disclosing it to other participants, thereby protecting privacy. Therefore, FL has been widely employed in various privacy-sensitive applications such as financial services [4], [5], smart healthcare [6], and recommender systems [7].

Vertical Federated Learning (VFL) is a significant category within FL that has garnered widespread attention. VFL considers a common scenario where parties share sample ID space but have disjoint sets of private features [8], [9], [10]. In this framework, the participant holding both sample labels and features is denoted as the active party, while those holding only sample features are referred to as passive parties. In the VFL context, participants generally involve one active party and multiple passive parties [11], [12]. During training, the active party distributes the homogeneous local model to the passive parties. Participants typically employ a homogeneous local model to generate embeddings of local feature datasets. Embedding means turning high-dimensional data into dense, compact vectors that keep important features, make computation faster, and protect privacy better. The active party uses the embedding to obtain predictions [13], [14], [15], [16]. Each participant contributes local knowledge from their local model to the aggregation in training global models [13], [14], [15], [16]. Typically, the active party distributes homogeneous local models to all passive parties, rather than allowing them to select heterogeneous local models autonomously based on local computational resources. This can degrade performance due to inefficient resource utilization and suboptimal adaptation to varying capabilities of passive parties. Furthermore, due to the absence of labels among passive parties, each round of model training necessitates frequent communication between the active and passive parties, which often incurs considerable communication overhead. That is to say, high communication overhead and heavy reliance on homogeneous local models are generally caused by a high dependency on the passive party's local knowledge in each global epoch.

Unfortunately, the reduced model performance resulting from the joint training of heterogeneous local models is a challenge in contemporary VFL. The local models of the passive party are mainly passively distributed by the active party, which leads to limitations in model selection. In existing

Received 20 June 2024; revised 16 December 2024; accepted 7 January 2025. Date of publication 16 January 2025; date of current version 7 March 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB2701300; in part by the National Natural Science Foundation of China under Grant U24B20146, Grant 62372044, and Grant 62232002; and in part by Beijing Municipal Science and Technology Commission Project under Grant Z241100009124008. The associate editor coordinating the review of this article and approving it for publication was Prof. Stefano Calzavara. *(Shuo Wang and Keke Gai are co-first authors.) (Corresponding author: Jing Yu.)*

Shuo Wang, Keke Gai, Zijian Zhang, and Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: 3220215214@bit.edu.cn; gaikeke@bit.edu.cn; zhangzijian@bit.edu.cn; liehuangz@bit.edu.cn).

Jing Yu is with the School of Information Engineering, Minzu University of China, Beijing 100086, China (e-mail: jing.yu@muc.edu.cn).

Digital Object Identifier 10.1109/TIFS.2025.3530700

VFL methods, such as SplitVFL [13], [14] and aggVFL [15], [16], the passive party usually relies on homogeneous models provided by the active party. The active party distributes models according to its own needs and resources without considering the hardware capabilities and computing resources of the passive party. This approach ignores the actual situation of the passive party, resulting in large differences in the applicability and efficiency of the model among different participants. In the real world, participants have different hardware capabilities and computing resources, which makes the use of a heterogeneity model not always optimal. For example, different medical institutions may have different computing capabilities and feature sets in the medical field, which complicates the collaborative training process. This heterogeneity may lead to poor performance and convergence of the global model [17], [18], [19], [20]. Therefore, eliminating negative impacts on the global model from heterogeneous local models of passive parties is a great challenge in VFL implementations.

High communication overhead is another challenge when implementing local multi-round training of participants without revealing active party's labels [21]. Existing methods are mainly grouped into two categories, which are data compression [8], [22], [23] and local multi-round training [24], [25]. Specifically, local data compression typically passively compresses data before initiating communications, so this manner barely reduces communication rounds. The reason is that the passive party does not hold label values, so it has to communicate with the active party to obtain the loss value at each training epoch. In addition, recent explorations on local multi-round training also probe that the implementation is impractical due to unrealistic assumptions issues [26], [27], [28] and label compromise [16], [29]. While local multi-round training methods effectively reduce the number of communication rounds, achieving local multi-round training without compromising label privacy remains a challenge.

To address the challenges above, we propose a novel scheme entitled *Practical Vertical Federated Learning with Heterogeneous Models via Representation Learning* (PraVFed), which aims to achieve collaborative training among participants with heterogeneous models while eliminating communication overheads. The primary principle of our scheme is to learn and aggregate the embedding values of passive participants' local features to eliminate the negative impacts on global models from heterogeneous models. Additionally, we employ blinding factors to protect the local embedding values of the passive party, transmitting blinded local embedding values to the active party. We develop a weighted aggregation-based method for strengthening the measurement of correlations between the local features held by passive participants and the global model. To avoid label privacy leakage and reduce communication rounds, we have proposed a perturbed multi-round training local pre-training model to obtain embedding values of passive parties' local features.

The main contributions are summarized as follows.

- Our work supports the implementation of heterogeneous models while considering the reduction of communication overhead in VFL. Local features of passive parties derive from aggregating values of blinded embedding, which lowers the negative impact made by the information of local heterogeneous models. We also propose a weighted embedding-based aggregation method to measure better the correlation between the global model and data features of passive parties; i.e., the correlation has a positive relationship with the global model.
- We propose a multi-round local pre-model training scheme with perturbation. The passive party obtains satisfied embedding values from adopting multi-round local pre-training models by using perturbed labels. This scheme not only protects the label privacy of the active party but also reduces communication overhead.
- We have conducted extensive experiments on classic datasets and neural networks to evaluate the performance of our scheme. Experiment results depict that the proposed PraVFed can successfully match the functional requirements of VFL and eliminate communication overhead. Specifically, our findings show that when the target accuracy is set at 60% under the CINIC10 dataset, the communication cost of PraVFed is reduced by 70.57% compared to the baseline method.

The rest of this paper is organized in the following order. We briefly review the related literature in Section II. Section III presents the preliminaries. Our proposed methodology is presented in Section IV, followed by the experience evaluations in Section V. Finally, we summarize this paper in Section VI.

II. Related Work

Model heterogeneity is a common topic in Horizontal Federated Learning (HFL) schemes [30]. Solutions for addressing model heterogeneity in HFL include knowledge distillation (KD) [20], [31], [32], [33] and Partial Training (PT) [19]. Typical KD-based methods extract knowledge from teacher models and add it to student models with heterogeneous model architectures; rather, PT-based methods extract sub-models from the global server model. An assumption of parties that independently train a complete local model is made by most existing FL methods. However, prior methods cannot satisfy VFL's needs since passive parties in VFL generally lack labels and cannot independently train a local model. Additionally, in VFL research, Castiglia et al. [34] proposed Flex-VFL, where participants with heterogeneous optimizers achieve global model training using a parallel block coordinate descent approach. This work focuses on the heterogeneity of local optimizers rather than the structural heterogeneity of passive parties' local models. Moreover, it assumes that all passive parties have copies of the labels, which does not align with real-world VFL scenarios where only one participant possesses the labels, while the other participants have disjoint feature sets. Therefore, there is currently a lack of research on the collaborative training of participants with heterogeneous local models in real-world VFL scenarios.

Communication overhead is a critical metric for evaluating the performance of VFL. Recent research has extensively focused on reducing this overhead. Address methods of communication include data compression and minimizing the number of communication rounds between parties. For instance, C-VFL [35] employs multiple local iterations by

the server and parties, periodically sharing compressed intermediate results to collaboratively train the model based on their respective features. Methods like TVFL [23] and LESS-VFL [22] further reduce communication overhead by selecting features from participants that are highly significant to the model.

However, these methods still require the exchange of embedding values between passive and active parties at each training step, resulting in frequent and substantial communication overhead. Recent advancements aim to reduce VFL communication frequency by allowing clients to perform multiple local updates per round. For example, CELU-VFL [24] minimizes cross-party communication rounds by caching and reusing outdated statistical data to estimate model gradients, eliminating the need to exchange temporary statistical data. Nonetheless, this approach is limited to two-party scenarios and does not scale to more than two parties. Alternatively, FedBCD [36] enables clients to update their local models for multiple steps using outdated gradients received from the server. Flex-VFL [34] allows each party to perform a variable number of local updates, within a timeout constraint for each round. However, Flex-VFL assumes all clients have label copies and receive local embeddings from others, facilitating independent local gradient computation for multi-step updates. In contrast, we propose a framework that supports heterogeneous models in VFL, allows for multiple rounds of local updates, and assumes only the active party holds the labels, while the other parties do not.

III. Preliminaries

A. Vertical Federated Learning

Vertical Federated Learning (VFL) is a specialized distributed machine learning framework that enables multiple organizations or entities to train a machine learning model collaboratively without sharing their private data. In VFL, the participating parties possess datasets that contain the same samples but different features. We consider a general VFL scenario of the real world involving an active party and multiple passive parties. The active party typically coordinates the training process and integrates contributions from all participating parties. The passive parties assist the active party in completing the final machine learning training by providing their unique features without sharing raw data. We summarize the notations used in this paper in Table I.

In this work, we denote the active party by l_1 and the passive parties by $\{l_k\}_{k=2}^{K}$. This work aims to facilitate passive parties' involvement in the active party's local model training while maintaining all participants' data locally for privacy protection. Efficiency is also considered in this work. We aim to enhance the efficiency of model training by eliminating communication overhead during the VFL training process. This work sets the active party's local model as the VFL's global model. The training problem of `PraVFed` is formulated by Equation (1).

$$\min_{\theta} \ell(\theta, D) :\triangleq \frac{1}{N}\sum_{i=1}^{N} L\left(\theta; \mathbf{x}_i, y_i\right). \tag{1}$$

TABLE I
Notations Table

Notation	Explanations
l_k	kth parties and l_1 denote active party
θ	Model parameters
PK	Public key
SK	Private key
CK	Share key
D	Training datasets
N	The total number of data samples
$\mathbf{x}_i$	ith data sample
y_i	the label of ith data sample
K	The total number of clients participating in training
$\{x_i\}_{l_k}$	Some of the local features owned by the l_kth client
$L(\cdot)$	The loss function
ϵ	Privacy budget of DP
δ	Probability bounds
H	Hash function
Δf	The L_2 sensitivity of f
Y	The labels of training samples
Y'	The noised labels of training samples
$\mathbf{g}$	The gradient
T_{l_k}	The training epochs of l_kth participant
E_{l_k}	The local embedding of l_kth participant
E_p	The global embedding of all passive parties
R_{l_k}	The prediction result of l_kth participant
w_{l_k}	The weight value of l_kth participant
r_{l_k}	The blinding factor

In Eq. (1), θ represents the model parameters to be trained that are owned by the active party. D denotes datasets; N denotes the total number of data samples; $\mathbf{x}_i = \bigcup_{k=1}^{K}\{x_i\}_{l_k}$ denotes ith data sample, y_i denotes the label of ith data sample; K represents the total number of clients participating in training; $\{x_i\}_{l_k}$ denotes some of the local features owned by the l_kth client; $L(\cdot)$ denotes the loss function.

B. Differential Privacy

In our work, DP is adopted for protecting perturbed labels. The active party injects noises that satisfy DP into labels, which ensures that the probability distributions of two random data are similar so that the attacker can hardly distinguish the current input dataset from D_1 and D_2. Thereby, the privacy of data l is protected.

Definition 1 (Differential Privacy (DP) [37]): A randomized function $\mathcal{W}: \mathcal{X} \to \mathcal{Y}$ satisfies (ϵ, δ) - DP only when $\forall$ adjacent data sets $\mathcal{D}, \mathcal{D}' \in \mathcal{X}$ and all possible output $y \in \mathcal{Y}$, it holds that

$$\Pr\left(\mathcal{W}\left(\mathcal{D}\right) \in \mathcal{Y}\right) \leq e^{\epsilon}\Pr\left(\mathcal{W}\left(\mathrm{D}'\right) \in \mathcal{Y}\right) + \delta \tag{2}$$

where ϵ represents the privacy budget, δ represents the probability bounds. Adjacent datasets $\mathcal{D}, \mathcal{D}'$ mean that only one of the samples differs.

Lemma 1 (DP of Laplace Mechanism): If the sensitivity of a function $f: X \to Y$, then the Gaussian mechanism $G(f) = f(D) + \mathcal{N}(0, \sigma^2)$ satisfies $\left(\alpha, \frac{\alpha \Delta f^2}{2\sigma^2}\right)$ - RDP, where $D \subseteq X$, Δf is the L_2 sensitivity of f defined by $\Delta f = \max_{D,D'} \|f(D) - f(D')\|_2$ with D and D' are adjacent datasets

C. Key Agreement

Key Agreement comprises three algorithms (*KA.param*, *KA.gen*, *KA.aggree*). The algorithm KA.param(o) generates a

public parameter pp, where o is the scheme's security parameter. Any passive party l_k can generate a private-public key pair according to the algorithm KA.gen$(pp) \rightarrow (PK_{l_k}, SK_{l_k})$. The algorithm KA.agree$(SK_{l_k}, PK_{l_j}) \rightarrow CK_{k,j}$ allows the passive party l_k to combine their private key SK_{l_k} with the public key PK_{l_j} of any passive party l_j (generated using the same pp) to obtain a private shared key $CK_{k,j}$ between passive party l_k and passive party l_j.

This work will use the Diffie-Hellman key agreement combined with a hash function. In particular, the algorithm KA.param$(o) \rightarrow (G', g, q, H)$ generates a group G' of prime order q, a generator g and a hash function H. The algorithm KA.gen$(G', g, q, H) \rightarrow (g, g^x)$ samples a random $x \leftarrow \mathbb{Z}_q$ as the secret key SK_{l_k}, and g^x as the public key PK_{l_k} of the l_kth passive party. The algorithm KA.agree$(SK_{l_k}, g^{SK_{l_j}}) \rightarrow CK_{k,j}$, where $CK_{k,j} = H((g^{SK_{l_j}})^{SK_{l_k}})$.

Passive party l_k and l_j generate any public-privacy key pairs using the algorithm KA.gen and the same parameters pp, which satisfies the Eq. (3).

$$KA.agree(SK_{l_k}, PK_{l_j}) = KA.agree(SK_{l_j}, PK_{l_k}). \tag{3}$$

For security, we want an adversary who cannot obtain the original information from the secret share generated by a public-private key pair. The Decisional Diffie-Hellman (DDH) assumption can guarantee the above security requirements in a semi-honest model. We define the DDH assumption below.

Definition 2 (Decisional Diffie-Hellman Assumption): Let $G(o) \rightarrow (G', g, q, H)$ be an efficient algorithm that generates a group G' of prime order q with generator g, as well as a function $H : \{0,1\}^* \rightarrow \{0,1\}^o$. Consider the following probabilistic experiment, parameterized by a PPT adversary M, a bit b, and a security parameter o:

1) $(G', g, q, H) \leftarrow G(o)$
2) $a \leftarrow \mathbb{Z}_q : A \leftarrow g^a$
3) $b \leftarrow \mathbb{Z}_q : B \leftarrow g^b$
4) *If* $b = 1$, $s \leftarrow H(g^{ab})$, *else* $s \leftarrow \{0,1\}^k$
5) $M(G', g, q, H, A, B, s) \rightarrow b'$
6) *Output 1 if* $b = b'$, *0 otherwise.*

The Decisional Diffie-Hellman (DDH) assumption can ensure that the shared secret $CK_{k,j}$ generated by any public-private key pair calculated by all attackers is indistinguishable from a uniform random string.

IV. Proposed Model

A. Design Goals

In this section, we present our design goals from different views. Our design goals are as follows.

- **Privacy:** To avoid the active party's label information leakage, we require the active party to inject noise in the label to satisfy (ϵ, δ)-DP. In addition, we require the passive party to add a blinding factor in the local representation to prevent the uploaded local representation from leaking the original data. PraVFed should be able to prevent private information from being leaked.
- **Accuracy:** The primary purpose of applying PraVFed is learning model accuracy. PraVFed should improve the learning accuracy by optimizing the model while ensuring the privacy of the local dataset.
- **Efficiency:** PraVFed aims to optimize model training efficiency by reducing communication overhead but introduces additional storage overhead. These additional costs mostly are caused by the inherent complexity of the training process. The overarching goal of PraVFed is to enhance model training efficiency while maintaining the incurred overhead within acceptable limits.

B. Threat Model

In VFL, all participants aim to train an optimal model while safeguarding local privacy collaboratively. We assume that all participants are honest but curious, which means that they adhere to the protocol but are interested in extracting additional information from the process. Thus, participants are not expected to engage in malicious behavior, e.g., injecting false data or disrupting training process. The active party efficiently manages the global model training, incorporating inputs from all passive parties and coordinating model updates, potentially leveraging intermediate data to enhance insights or achieve a competitive edge. Conversely, the passive party accurately performs local pre-training and embedding, using its resources to generate high-quality embeddings while potentially inferring the active party's label information.

In this work, our assumption does not consider external attackers and excludes the possibility of malicious tampering with communication messages. We assume that a secure communication channel is established between the active and passive parties, ensuring that all transmitted data is encrypted and protected from interception or tampering by external entities. This secure channel guarantees that external attackers cannot detect communication information during training.

C. Overview

We present an overview of the PraVFed in Fig. 1. Our scheme aims to address the challenge of low accuracy and high communication overhead in heterogeneous parties by employing local embedding extraction and weight aggregation to achieve a practical VFL scheme. Our scheme comprises K entities that consist of an active party and $K-1$ passive parties. The active party primarily executes three modules: label perturbation, weighted aggregation, and model training. The passive party utilizes local representation learning to assist the active party in completing the training of the global model. Two modules primarily performed by passive parties are local model pre-training (refer to IV-D1) and local embedding (refer to IV-D2) during the training process.

We implement PraVFed for the active-passive model. During training, the active party injects the noise into the true labels and sends the perturbed labels to all passive parties. To reduce the communication overhead, each passive party trains a pre_model locally using local features and perturbed labels. We will obtain passive parties' local embedding using pre_model while keeping privacy. Then, each passive party sends the local embedding to the active party, which assists the active party in training the global model. In each round

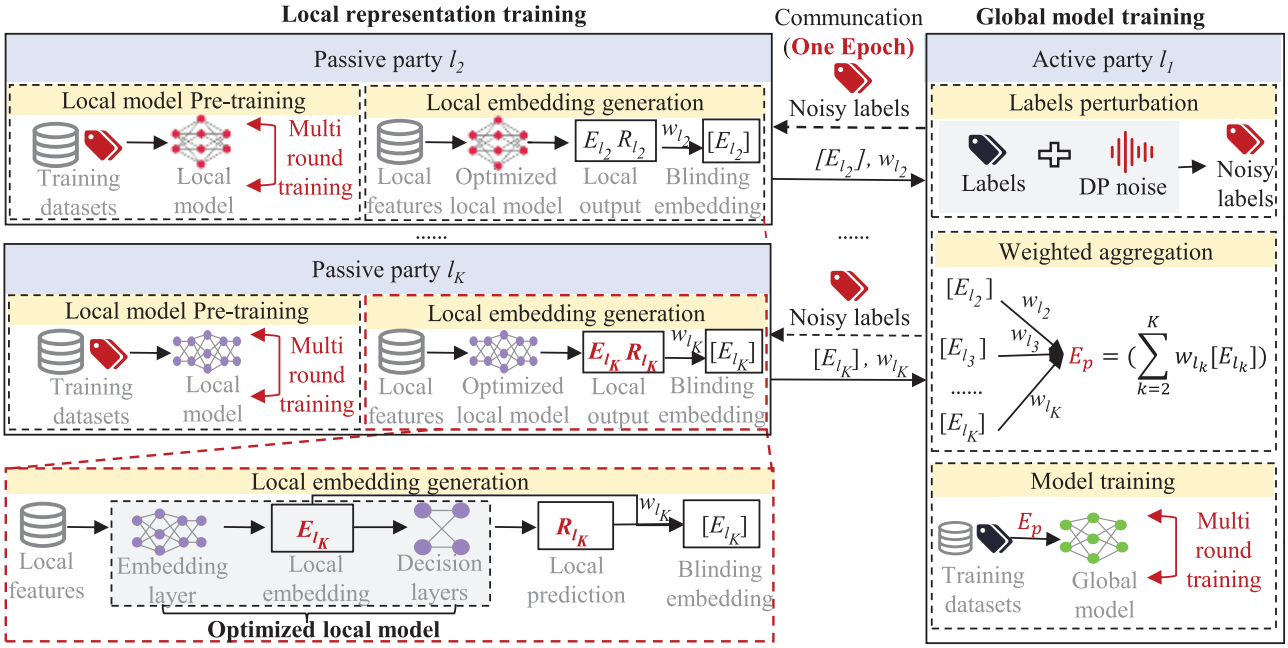


Fig. 1. An overview of the PraVFed in the heterogeneous models setting. Each participant owns the heterogeneous local model and a partial subset of features. The active party owns sample labels, partial features, and heterogeneous local models. K represents the number of participants in PraVFed.

of global model communication, the active party first obtains local embedding values using local features. Then, it uses an aggregation algorithm to obtain global embedding values. Finally, the global embedding values are used to complete the global model's training. We will provide a detailed description of this approach in the following sections.

D. Local Representation Training

For obtaining the passive parties' local data features while keeping data private, we propose a local representation learning method. The local representation learning approach has three key components. The *local model pre-trained* is used to train the passive side's local model. The *local embedding generation* produces embedding values using the pre-trained model and local data of the passive party. The *local embedding blinding* enhances the security of local data by incorporating blinding elements into the local embedding. Next, we will provide a comprehensive overview of the aforementioned three fundamental parts.

1) Local Model Pre-Trained: The passive party conducts multiple rounds of independent local model training to reduce the communication rounds between the passive parties and the active party. It also facilitates the acquisition of an enhanced pre-trained model. Additionally, we obtain superior embedding for local features of the passive party deriving from adopting pre-trained models.

Algorithm 1 shows the process of the local model pre-trained on passive parties. All passive parties train local pre-trained models in parallel. In particular, the local model pre-training process of l_kth passive party includes the following steps. First, the l_kth passive party initializes the local model $\theta_{l_k}^0$, and the learning rate is $\eta_{l_k}^0$. Then, the passive party utilizes local features and disturbance labels Y' to compute the loss

Algorithm 1 Local Model Pre-Trained (LP)

Require: Local training features D_{l_k}, Local epoch T_{l_k}, Passive parties l_k, Number of samples N, Noise labels Y', Batch size B_{l_k}

Ensure: Pre-trained model θ_{l_K}

```
Initialize local model θ^0_{l_k}
for each epoch t = 0, ..., T_{l_k} − 1 do
    B, Y' ← dividing D_{l_k}, Y' into batches of size B_{l_k}.
    for each batch (b, y') ∈ (B, Y') do
        g^(t) ← ∇_{θ^(t)_{l_k}} L^(t) (θ^(t)_{l_k}, b, y')
        θ^(t+1)_{l_k} = θ^(t)_{l_k} − η^(t)_{l_k} g^(t)
    end for
end for
θ_{l_k} ← θ^(T_{l_k})_{l_k}
return: θ_{l_k}
```

function during local training, which can be expressed by Equation (4).

$$L^{(t)}\left(\theta_{l_k}, D_{l_k}\right) \leftarrow \frac{1}{2N}\left\| f^{(t)}\left(\theta_{l_k}, D_{l_k}\right) - Y' \right\|^2 \tag{4}$$

The local model is updated by using gradient descent (refer to Equation (5)), where η_{l_k} is the learning rate and $\mathbf{g}$ is the gradient.

$$\theta_{l_k}^{(t+1)} = \theta_{l_k}^{(t)} - \eta_{l_k}^{(t)} \mathbf{g}^{(t)} \tag{5}$$

After preceding T_{l_k} training epochs, we will obtain the trained local pre-trained model θ_{l_k}. All other passive parties follow the steps above to set up local model pre-training. Finally, we eventually obtain the local optimal pre-trained model of the passive party.

2) Local Embedding Generation: To reduce the negative effects of the local heterogeneous models of passive parties on the global model of the active party, we adopted a strategy inspired by previous studies [38], [39]. The pre-trained local network of the passive side is divided into an embedding network (e.g., embedding layer) and a prediction network (e.g., decision layers). **The embedding layer** embeds some features owned by every participant in the same embedding space. The embedding layer of the l_kth participant is $h(\theta_{l_k})$. We denote $E_{l_k} = h(\theta_{l_k}, D_{l_k})$ as the embedding of D_{l_k} for l_kth participant. **The decision layers** are the predicting result for features D_{l_k} based on the unsupervised learning task. We represent the prediction result by using $R_{l_k} = p(\theta_{l_k}, D_{l_k})$.

3) Local Embedding Blinding: The embedding E_{l_k} contains valuable local feature information from the passive party. Therefore, the passive party utilizes the embedding values to assist the active party in completing the training of the global model. However, an honest but curious active party may attempt to deduce the passive party's local features from the embeddings E_{l_k}. To project the passive party's local sample features, we introduce a blinding factor to the embeddings E_{l_k}. To generate blinding factors, each passive party l_k first initializes a pair of public-private keys (PK_{l_k}, SK_{l_k}) according to the algorithm KA.gen (Section III-C). Subsequently, the passive party l_k distributes the public key PK_{l_k} and the aggregation weight w to other passive parties. And, passive party l_k computes the shared key $CK_{k,j} = H((PK_{l_j})^{SK_{l_k}})$ by the algorithm KA.agree (Section III-C), where $k, j \in [1, K]$, $j \neq k$, and $H(\cdot)$ represents a secure hash function resistant to collusion, capable of transforming strings of any length into elements in $\mathbb{Z}_p$.

Eq. (6) defines the mathematical relationship between the shared keys.

$$\begin{aligned} CK_{k,j} &= H((PK_{l_j})^{SK_{l_k}}) = H((g^{SK_{l_j}})^{SK_{l_k}}) \\ &= H((g^{SK_{l_k}})^{SK_{l_j}}) = H((PK_{l_k})^{SK_{l_j}}) = CK_{j,k} \end{aligned} \tag{6}$$

Each passive party l_k generated the blinding factor r_{l_k}, as shown in Eq. (7).

$$r_{l_k} = \sum_{j \in [1,k], j \neq k} (-1)^{k>j} w_{l_j} \times H(CK_{k,j}) \tag{7}$$

where $(-1)^{k>j} = -1$ if $k > j$. Particularly, the sum of all blinding factors $(\sum_{k \in [1,K]} w_{l_k} \times r_{l_k})$ is 0.

Each passive party l_k introduces the blinding factors to the local embedding and obtains the privacy-protected embedding values$[E_{l_k}] = E_{l_k} + r_{l_k}$. Then, each passive party l_k transmits the embedding values $[E_{l_k}]$ with blinding to the active party. Additionally, we also transmit the weight w_{l_k} of the passive party's pre-trained model to the active party to explore the impact of local sample features on model training. The weight is employed to assist the active party in assessing the importance of the global aggregation performed by the passive party.

E. Global Model Training

The active party's training procedure consists of three essential phases: *label perturbation*, *weight aggregation*, and *model training*. The active party has to first inject noise into the actual labels before sending them to the passive party to help it extract local features. Second, to achieve global embedding, the active party aggregates the local embedding values of the passive party using the weight aggregation method. Finally, the global model is trained by the active party using the global embedding value.

Algorithm 2 PraVFed Training

Require: Datasets D_{l_k}, Local epoch T_k, clients $k = 1, \ldots, K$, Number of samples N, Epoch T, learning rate η, Labels Y, Privacy budget ϵ_{l_k}, sensitivity $\triangle f$

Ensure: Excellent θ

1: **Active party**
2: $Y' \leftarrow Y + Lap(\triangle f/\epsilon)$.
3: Sent the Y' to the passive parties.
4: **Passive parties parallel**
5: **for** $l_k \in [2, K]$ **do**
6: $\theta_{l_k} \leftarrow$ LP(D_{l_k}, T_{l_k}, l_k,N,Y') //refer to Algorithm 1
7: $E_{l_k} \leftarrow h(\theta_{l_k}, D_{l_k})$
8: $R_{l_k} \leftarrow p(\theta_{l_k}, E_{l_k})$
9: $ACC_{l_k} =$ L(R_{l_k}, Y')
10: Sent ACC_{l_k} to other passive parties
11: $w_{l_k} = \frac{ACC_{l_k}}{\sum_{k=2}^{K} ACC_{l_k}}$
12: Calculate the blinding factor r_{l_k}
13: $[E_{l_k}] \leftarrow E_{l_k} + r_{l_k}$
14: Send $[E_{l_k}]$, w_{l_k} to active party
15: **end for**
16: **Active party**
17: Received local embedding $[E_{l_2}], [E_{l_3}], \ldots, [E_{l_K}]$
18: Calculate the global embedding E_p of the passive party according to Equation (9)
19: Initialize global model θ^0
20: **for** $t \in [1, T]$ **do**
21: $E_a = h(\theta^0, D_{l_1})$
22: $E = E_a + E_p$
23: $R = p(\theta^0, E)$
24: $\mathbf{g}^{(t)} \leftarrow \nabla L^{(t)}(R, Y)$
25: $\theta^{(t+1)} = \theta^{(t)} - \eta^{(t)} \mathbf{g}^{(t)}$
26: **end for**
27: **return:** $\theta \leftarrow \theta^T$

1) Labels Perturbation: The active party injects noise into the local labels that satisfy DP and subsequently sends the perturbed labels to the passive party, aiding them in completing the local pre-training process. This reduces the need for extensive communication between the active and passive parties. The active party computes the noise to be added and perturbs local labels (Line 1-3 in Algorithm 2):

$$Y' \leftarrow Y + Lap(\Delta f/\epsilon) \tag{8}$$

where *Lap* represents Laplace distribution, Δf denotes sensitivity, ϵ represents the privacy budget.

2) Weight Aggregation: Existing aggregation-based VFL cannot provide enough effective information for each VFL client with heterogeneous models. Fortunately, the embedding layer of the heterogeneous network can obtain the same embedding space with the same label features. We effectively

obtain the local features of clients by weight aggregating the local embedding.

This work implements the weight embedding aggregation in the active party. The active party receives the local embedding $\{[E_{l_k}]\}_{k=2}^{K}$ transmitted by the passive party using the pre-trained model and the weight value $\{w_{l_k}\}_{k=2}^{K}$. The global embedding E_p of all passive parties is aggregated by the active party, utilizing a weight generated by Equation (9). The aggregation method makes it possible to efficiently gather the passive party's local feature knowledge.

$$\begin{aligned} E_p &= \sum_{k=2}^{K}(w_{l_k} \times [E_{l_k}]) \\ &= \sum_{k=2}^{K}\{(w_{l_k} \times E_{l_k}) + (w_{l_k} \times r_{l_k})\} = \sum_{k=2}^{K}(w_{l_k} \times E_{l_k}) \end{aligned} \tag{9}$$

3) Training Process: The global model training process for implementing PraVFed is shown in Algorithm 2. We use l_1 to represent the active party for convenience. Clients only have part of the features and need the assistance of other passive parties during the training process. The training process of PraVFed mainly includes the following steps. *Step 1* (line 1 to line 7 of the Algorithm 2) According to Algorithm 1, the passive parties employ unsupervised learning to train the local pre-trained model. The passive party then uses the locally pre-trained model to obtain the embedding and local prediction values and sends them to the active party. *Step 2* (lines 9 to 13) The active party implements a weighted aggregation method to integrate the local Embedding values of the passive party. The active party initializes the global model θ^0. *Step 3* (lines 14 to 21) The active party trains the global model. We also split the active party's global network into an embedding network and a prediction network. Primarily, the active party obtains the local feature's embedding value E_a. Second, we add up the passive parties' embedding values E_p to get E and then calculate the global model's result. Lastly, the active party obtains the model loss value using the loss function and uses the gradient descent method to update the global model. We can choose different loss functions according to the task requirements. After T training rounds, we will obtain global models with excellent performance.

F. Theoretical Analysis

1) Security Analysis of Blinding Factor: In PraVFed, we utilize blinding factors to mask the passive party's local embedding and preserve its local features. If the blinding factor is secure, then other parties will not be able to distinguish the true embedding from the obfuscated embedding, and as a result, other parties will not be able to infer the true features. Therefore, we provide security proof demonstrating the safety of obtaining the blinding factor.

Proof: In PraVFed, the active party owns the public key PK_l for each passive party l, along with the corresponding local embedding with blinding factors and aggregate weight provided by the passive parties during the training process. We demonstrate that passive parties cannot access the shared key. Therefore, our method guarantees that they cannot gain the true local embedding of other passive parties. The Computational Diffie-Hellman (CDH) problem ensures that given g^a and g^b, the probability of computing the g^{ab} is negligible. Therefore, when provided with access to all public keys $PK_l = g^{SK_l}$, the semi-honest passive party l is unable to infer the shared key $CK_{k,j} = H((g^{SK_{l_j}})^{SK_l})$, where $\forall k, j \in [1, K], j \neq k \neq K$. Each party only possesses its private key, and even in the event of collusion among multiple semi-honest parties, they cannot retrieve the shared key between the other parties. Furthermore, semi-honest parties cannot obtain the true local embedding of other parties. Therefore, they will not be able to infer the original features of other parties from the true local embedding. ■

This proof further illustrates that during the training process, the local features of all passive parties will not be leaked, ensuring the privacy of the features. Therefore, the PraVFed method ensures that the honest-but-curious participants cannot obtain true local embedding with blinding factors from global embedding and thus cannot infer the raw features. The PraVFed method effectively protects the raw features of the passive parties.

2) Privacy Analysis of Labels: To safeguard the privacy of the active party's label, the active party generates a perturbed label Y' by adding random noise to the original label Y. The passive party pre-trains the model locally by receiving perturbed labels from the active party. We prove below that perturbed label Y' satisfies differential privacy.

Proof: Assume the queries from Y and Y' are both z for any adjacent datasets D and D'. We obtain the Equation 10 and Equation 11 by using $Y'(D) = Y(D) + Lap\left(\frac{\Delta f}{\epsilon}\right)$.

$$\begin{aligned} \Pr(\mathrm{Y'(D)} = z) &= \Pr\left(\mathrm{Y(D)} + \mathrm{Lap}\left(\frac{\Delta \mathrm{f}}{\epsilon}\right) = z\right) \\ &= \Pr\left(\mathrm{Lap}\left(\frac{\Delta \mathrm{f}}{\epsilon}\right) = z - \mathrm{Y(D)}\right) \end{aligned} \tag{10}$$

$$\begin{aligned} \Pr(\mathrm{Y'(D')} = z) &= \Pr\left(\mathrm{Y(D')} + \mathrm{Lap}\left(\frac{\Delta \mathrm{f}}{\epsilon}\right) = z\right) \\ &= \Pr\left(\mathrm{Lap}\left(\frac{\Delta \mathrm{f}}{\epsilon}\right) = z - \mathrm{Y(D')}\right) \end{aligned} \tag{11}$$

Using the probability density function of the Laplace mechanism, Equations 10 and 11 can be reconstituted as the Equation 12.

$$\begin{aligned} \Pr(\mathrm{Y'(D)} = z) &= \frac{\epsilon}{2\Delta \mathrm{f}} \exp\left(\frac{-\epsilon\,|z - \mathrm{Y(D)}|}{\Delta \mathrm{f}}\right) \\ \Pr(\mathrm{Y'(D')} = z) &= \frac{\epsilon}{2\Delta \mathrm{f}} \exp\left(\frac{-\epsilon\,|z - \mathrm{Y(D')}|}{\Delta \mathrm{f}}\right) \end{aligned} \tag{12}$$

The Equation 13 is obtained by deriving the ratio of $\Pr(\mathrm{Y'(D)} = z)$ and $\Pr(\mathrm{Y'(D')} = z)$. In addition, we know that the sensitivity $\Delta f = |Y(D) - Y(D')|$

$$\begin{aligned} \frac{\Pr(\mathrm{Y'(D)} = \mathrm{z})}{\Pr(\mathrm{Y'(D')} = \mathrm{z})} &= \frac{\frac{\epsilon}{2\Delta f} exp\left(\frac{-\epsilon|z-Y(D)|}{\Delta f}\right)}{\frac{\epsilon}{2\Delta f} exp\left(\frac{-\epsilon|z-Y(D')|}{\Delta f}\right)} \\ &= exp\left(\frac{\epsilon(|z - Y(D)| - |z - Y(D')|)}{\Delta f}\right) \end{aligned}$$

$$\leq exp\left(\frac{\epsilon|Y(D)-Y(D')|}{\Delta f}\right)$$
$$= exp(\epsilon) \quad (13)$$

Thus, we can achieve the following Equation 14:

$$\Pr\left(\mathrm{Y}'\left(\mathcal{D}\right)\in\mathcal{Y}\right)\leq \mathrm{e}^{\epsilon}\Pr\left(\mathrm{Y}'\left(\mathrm{D}'\right)\in\mathcal{Y}\right)+0 \quad (14)$$

■

According to the derivation above, the perturbed labels Y' generated by injecting random noise into the original label Y satisfies $(\epsilon, 0)$-DP. DP protected data l by adding random noises to the probability distribution of two random data so the attacker was unable to determine which was D_1 or D_2. Therefore, the passive party cannot obtain the real label from the perturbed label, thereby protecting the privacy of the active party's labels.

V. Experiments Evaluation

A. Experiment Configuration

1) Datasets: To evaluate our method, we run experiments on four image datasets and four tabular datasets according to the existing scheme [16], [35], [40], [41].

- *Image Datasets.* We evaluate our method using four commonly used image datasets: two relatively simple datasets, MNIST and FMNIST, and two more complex datasets, CIFAR10 and CINIC10. **MNIST** [42] is a picture data set of handwritten digits, including 60,000 training data sets and 10,000 test data sets. **Fashion-MNIST (FMNIST)** [43] is a frontal image of commodities covering 70,000 grey levels from 10 categories, including 60,000 training datasets and 10,000 test data sets. **CIFAR10** [44] is a collection of color pictures that was divided into 10 classes. **CINIC10** [45] is an extended version of the CIFAR10 dataset, created by augmenting it with ImageNet images. It contains 90,000 images across both training and testing sets, with ten distinct classes. The training set of CINIC10 is 1.8 times larger than that of CIFAR10, providing a more robust dataset for performance evaluation.
- *Tabular Datasets.* Additionally, we evaluate our method using four real-world tabular datasets, which reflect practical applications in various domains. **Adult Income** [46] is a classification dataset derived from census data used to predict whether an individual's annual income exceeds 50,000 USD. It contains a total of 48,842 samples, each with 14 features, including age, occupation, and education level. **Breast Cancer** [47] is a classification dataset used to differentiate between benign and malignant breast tumors. It contains 668 samples, each with 10 features. **Credit Card** [48] is used to predict credit card fraud or payment defaults. It contains 711 samples, each with 29 features. **Diabetes** [49] is used to predict whether a patient will develop diabetes. It includes physiological data from 442 patients, along with their health outcomes one year later. Each sample contains 10 features, such as age, gender, and body mass index.

To adapt to the scenario of VFL, for image datasets, we divide each sample into K parts along the feature dimension, as is common in existing work [16], [40]. For tabular datasets, we divide all features of each sample into K parts, consistent with prior work [41]. Additionally, we randomly select 80% of the samples for the training dataset and 20% for the test dataset The training dataset is utilized to train the global model, while the test dataset is employed to evaluate the accuracy of the PraVFed method.

2) Models: We selected different models in terms of the characteristics of each dataset [41], [50]. For simple image datasets, we use a heterogeneous model composed of common *Multi-Layer Perceptron(MLP)*, *Convolutional Neural Network (CNN)*, *LeNet* architectures. For more complex image datasets, we use various versions of ResNet, including ResNet-18, ResNet-34, and ResNet-50, to form a heterogeneous model. For tabular datasets, we employ the MLP model for training. The specific model configurations are as follows.

- *MLP* is a three-layer fully connected network, including the input, hidden, and output layers. The input layer maps dim_{in} to 50 hidden layers, followed by a dropout layer to prevent overfitting. The hidden layer maps 50 units to dim_{out}, representing the number of output classes. We apply ReLU activations to the hidden layers and use softmax at the output to generate probabilities, optimizing multi-class classification with cross-entropy loss.
- *CNN* consists of two convolutional layers and two fully connected layers. The first convolutional layer takes 1 input channel and produces 64 output channels, while the second layer takes 64 input channels and reduces them to 32 output channels. Both layers employ 1×1 convolutional filters. The fully connected layers project the extracted features into a 50-dimensional space, followed by a final projection to dim_{out} output classes. We apply ReLU activation functions to all intermediate layers and use a softmax function at the output layer to generate class probabilities, with the model optimizing multi-class classification using the cross-entropy loss function.
- *LeNet* consists of three convolutional layers, one pooling layer, and three fully connected layers.
- *ResNet* is a deep residual network. ResNet adds the input directly to the output by setting up an identity mapping in the residual block and using the activation function to transform linear. In addition, ResNet also uses global average pooling and fully connected layers as the final classifier. We choose different versions of ResNet18, ResNet34, ResNet50, and ResNet101 in PraVFed.
- *MLP for Tabular datasets* is a three-layer fully connected network, consisting of the input, hidden, and output layers. The input layer maps $input\_dim$ to 64 units in the first hidden layer, followed by a dropout layer to prevent overfitting. The second hidden layer maps these 64 units to 32 units. Finally, the output layer maps the 32 units to $output\_dim$, representing the number of output classes, with a softmax activation function applied to generate class probabilities.

During the training process, participants randomly selected a network, and multiple participants formed heterogeneous

TABLE II

COMPARISON OF VFL METHODS ON FOUR IMAGE DATASETS AND LOCAL MODELS ($\theta_1, \theta_2, \theta_3$) UNDER HOMOGENEOUS AND HETEROGENEOUS MODELS. THE CENTRAL METHOD AND THE STANDALONE METHOD ACT AS THE HIGHER BASELINE AND LOWER BASELINE, RESPECTIVELY. $\pm$ DENOTES THE STANDARD DEVIATION

Datasets	Methods	Testing Accuracy (%)					
		Homogeneous Models			Heterogeneous Models		
		θ_1	θ_2	θ_3	θ_1	θ_2	θ_3
MNIST	Central	99.51 $\pm$ 0.22	99.67 $\pm$ 0.08	98.02 $\pm$ 0.38	99.51 $\pm$ 0.22	99.67 $\pm$ 0.08	98.02 $\pm$ 0.38
	Standalone	23.48 $\pm$ 0.36	35.57 $\pm$ 0.14	31.79 $\pm$ 0.10	23.48 $\pm$ 0.36	35.57 $\pm$ 0.14	31.79 $\pm$ 0.10
	Pyvertical	92.68 $\pm$ 1.14	94.18 $\pm$ 1.16	91.77 $\pm$ 0.70	92.35 $\pm$ 1.02	93.07 $\pm$ 1.41	93.00 $\pm$ 1.09
	C_VFL	89.71 $\pm$ 1.65	94.72 $\pm$ 1.74	95.33 $\pm$ 4.60	94.78 $\pm$ 0.45	95.47 $\pm$ 0.47	94.73 $\pm$ 0.42
	Agg_VFL	91.72 $\pm$ 2.08	86.03 $\pm$ 1.41	95.71 $\pm$ 2.39	82.15 $\pm$ 2.42	85.62 $\pm$ 1.28	93.65 $\pm$ 2.61
	PraVFed (Our)	**96.37 $\pm$ 1.62**	**95.34 $\pm$ 1.00**	**95.83 $\pm$ 1.06**	**95.35 $\pm$ 0.35**	**96.56 $\pm$ 0.20**	**95.56 $\pm$ 0.05**
FMNIST	Central	97.82 $\pm$ 0.34	95.36 $\pm$ 0.78	89.23 $\pm$ 0.41	97.82 $\pm$ 0.34	95.36 $\pm$ 0.78	89.23 $\pm$ 0.41
	Standalone	65.55 $\pm$ 1.41	76.24 $\pm$ 1.04	70.40 $\pm$ 2.37	65.55 $\pm$ 1.41	76.24 $\pm$ 1.04	70.40 $\pm$ 1.17
	Pyvertical	72.62 $\pm$ 1.63	76.34 $\pm$ 1.05	73.86 $\pm$ 1.10	82.98 $\pm$ 1.74	77.59 $\pm$ 1.63	74.15 $\pm$ 1.15
	C_VFL	83.92 $\pm$ 1.68	84.32 $\pm$ 1.84	84.53 $\pm$ 1.05	82.97 $\pm$ 0.77	84.32 $\pm$ 1.44	83.98 $\pm$ 0.72
	Agg_VFL	84.37 $\pm$ 1.88	84.68 $\pm$ 1.41	84.37 $\pm$ 1.05	55.04 $\pm$ 2.82	60.11 $\pm$ 1.28	42.67 $\pm$ 1.51
	PraVFed (Our)	**85.07 $\pm$ 1.90**	**85.39 $\pm$ 1.06**	**85.15 $\pm$ 0.91**	**83.76 $\pm$ 0.07**	**87.41 $\pm$ 1.18**	**85.12 $\pm$ 0.10**
CIFAR10	Central	94.17 $\pm$ 0.29	95.21 $\pm$ 0.09	96.01 $\pm$ 0.06	94.17 $\pm$ 0.29	95.21 $\pm$ 0.09	96.01 $\pm$ 0.06
	Standalone	61.22 $\pm$ 0.45	61.69 $\pm$ 0.37	63.39 $\pm$ 0.72	61.22 $\pm$ 0.45	61.69 $\pm$ 0.37	63.39 $\pm$ 0.72
	Pyvertical	72.36 $\pm$ 0.09	74.87 $\pm$ 0.37	75.56 $\pm$ 0.09	72.94 $\pm$ 0.21	74.87 $\pm$ 0.29	75.07 $\pm$ 0.26
	C_VFL	74.93 $\pm$ 0.25	71.99 $\pm$ 1.02	64.22 $\pm$ 1.18	63. 30 $\pm$ 1.02	65.99 $\pm$ 1.06	65.70 $\pm$ 0.56
	Agg_VFL	64.42 $\pm$ 0.34	65.14 $\pm$ 0.19	63.76 $\pm$ 0.19	64.30 $\pm$ 0.21	64.62 $\pm$ 0.18	65.37 $\pm$ 0.28
	PraVFed (Our)	**83.75 $\pm$ 0.99**	**81.77 $\pm$ 0.70**	**81.52 $\pm$ 1.17**	**82.87 $\pm$ 1.25**	**83.38 $\pm$ 0.13**	**82.89 $\pm$ 0.30**
CINIC10	Central	80.47 $\pm$ 0.23	76.64 $\pm$ 0.14	81.07 $\pm$ 0.78	80.47 $\pm$ 0.23	76.64 $\pm$ 0.14	81.07 $\pm$ 0.78
	Standalone	52.56 $\pm$ 0.78	53.67 $\pm$ 0.35	54.16 $\pm$ 0.57	52.56 $\pm$ 0.78	53.67 $\pm$ 0.35	54.16 $\pm$ 0.57
	Pyvertical	60.67 $\pm$ 0.12	62.93 $\pm$ 0.07	64.12 $\pm$ 0.14	60.83 $\pm$ 0.05	62.90 $\pm$ 0.02	63.26 $\pm$ 0.13
	C_VFL	60.03 $\pm$ 0.17	62.32 $\pm$ 0.17	62.33 $\pm$ 0.18	62.62 $\pm$ 0.13	63.19 $\pm$ 0.97	63.48 $\pm$ 0.10
	Agg_VFL	61.41 $\pm$ 0.14	58.56 $\pm$ 0.16	58.36 $\pm$ 0.16	62.80 $\pm$ 0.10	59.21 $\pm$ 0.80	60.46 $\pm$ 0.97
	PraVFed (Our)	**62.90 $\pm$ 0.03**	**64.76 $\pm$ 0.04**	**66.30 $\pm$ 0.02**	**64.79 $\pm$ 0.02**	**69.23 $\pm$ 0.06**	**67.47 $\pm$ 0.02**

models for collaborative training. Moreover, in the homogeneous scenario, all clients use the same local model architecture.

3) Baseline Methods: To evaluate the performance of PraVFed, we compare it with baseline methods. We benchmark PraVFed against the methods.

- *Central:* Centralized training means one organization has all training datasets and trains the model by itself, which serves as the higher performance bound.
- *Standalone:* The active party independently trains its local model using the local dataset, which serves as a lower-performance bound.
- *Pyvertical [51]* describes a collaborative approach where one active party and multiple passive parties jointly train a global model. This method involves dividing the network, with each passive party holds a *homogeneous model*.
- *C_VFL [35]*: In C_VFL, each passive party passively has a *homogeneous local model*, and the intermediate results obtained from the local model are compressed and sent to the active party for aggregation.
- *Agg_VFL [52]* refers to each passive party having a complete local model and sending the prediction results of the local model to the active party, who completes the aggregation.

4) Implementation Details: The PraVFed and baseline methods are implemented using the PyTorch 1.10.0 [53] framework. We employ four clients in PraVFed, with one party assuming an active role and the remaining three adopting passive roles. To simulate the VFL scenario, we follow the method described in [16] and [40]. We assume the active party ownership of the labels assigned to all samples. All clients collaboratively train local heterogeneous models during the training process, ensuring alignment of the training samples. In order to balance privacy with model accuracy, we adjusted the training hyperparameters for each dataset based on the findings from the ablation experiments. Specifically, for simple image datasets, we configured the epochs to 20 and allocated a privacy budget of 1.0. For more complex image datasets, we set the epochs to 50, with the privacy budget increased to 2.0. For tabular datasets, we chose the epochs to 20, paired with a privacy budget of 0.5. The initial learning rate of the model is set to 0.01, which gradually decreases during training. Additionally, we repeated the training process five times with different random seeds to obtain the test accuracy and standard deviation for all experiments.

B. Evaluation on Model Accuracy

In this section, we evaluate the model accuracy of our method on differential datasets and models. In addition, we fairly compare our proposed method with other baseline VFL methods on model testing accuracy under the same experimental setting. Table II reports the testing accuracy of the active party's global model under homogeneous and heterogeneous local models (θ_1, θ_2, θ_3), respectively. The homogeneous model methods present that all the clients own the same model structure. On the other hand, the local heterogeneous model approach shows the accuracy of the active party's global model when all passive parties have heterogeneous local models.

Table II demonstrates that our PraVFed exhibits superior training performance compared to other baseline methods. For instance, when considering CINIC10 datasets to train

TABLE III

COMPARISON OF VFL METHODS ON FOUR CLASSIC TABULAR DATASETS. THE "-" INDICATES THAT THE CENTRAL AND STANDALONE METHODS DO NOT REQUIRE CLIENT COMMUNICATION, SO THERE IS NO COMMUNICATION OVERHEAD

	Testing Accuracy (%) / Communication Cost (MB)			
	Adult Income	Breast Cancer	Credit Card	Diabetes
Central	85.95 / -	98.07 / -	99.98 / -	78.88 / -
Strandalone	78.54 / -	95.17 / -	99.71 / -	63.87 / -
Pyvertical	84.15 / 6105	96.49 / 107	99.93 / 13350	74.67 / 143
C_VFL	83.83 / 4578	94.56 / 32	99.82 / 5340	75.32 / 65
Agg_VFL	83.71 / 95	95.61 / 2	99.92 / 843	67.27 / 2
PraVFed	84.71 / 248	96.49 / 4	99.93 / 1502	76.35 / 5

TABLE IV

COMMUNICATION COST OVER FOUR IMAGE DATASETS UNDER HETEROGENEOUS MODELS. THE "-" INDICATES THAT THE METHOD DID NOT ACHIEVE THE TARGET ACCURACY AND THUS RESULTED IN NO COMMUNICATION OVERHEAD

Datasets	Methods	Max Accuracy Reached	Target accuracy (%) / Comm Cost (MB)
MNIST	Pyvertical	93.07% ± 1.41%	85 % / 1464.84MB
	C_VFL	95.47% ± 0.47%	85 % / 3662.11MB
	Agg_VFL	85.62% ± 1.28%	85% / 54.97MB
	PraVFed	96.56% ± 0.20%	85% / 329.59MB
FMNIST	Pyvertical	77.59% ± 1.63%	70% / 2929.68MB
	C_VFL	84.32% ± 1.44%	70% / 3662.11MB
	Agg_VFL	60.11% ± 1.28%	70%/ -
	PraVFed	87.41% ± 1.18%	70%/ 329.58MB
CIFAR10	Pyvertical	74.87% ± 0.29%	60% / 292.96 GB
	C_VFL	65.99% ± 1.06%	60% / 732.42GB
	Agg_VFL	64.62% ± 0.18%	60% / 12.06GB
	PraVFed	83.38% ± 0.13%	60% / 73.33GB
CINIC10	Pyvertical	62.90% ± 0.02%	60% / 896.48GB
	C_VFL	63.19% ± 0.97%	60% / 2636.32GB
	Agg_VFL	59.21% ± 0.80%	60% / -
	PraVFed	69.23% ± 0.06%	60% / 263.83GB

the model θ_2 under homogeneous and heterogeneous types, our method shows a test accuracy improvement of 2.9% and 6.9% compared to the highest accuracy baseline method. This is because the intermediate or predicted results aggregated by existing VFL methods contain more local model information. Therefore, the results in Table II showed that when the local model of the passive party is homogeneous or heterogeneous, PraVFed could obtain similar model training performance. This finding shows that PraVFed supported the training of heterogeneous models without reducing model accuracy, expanding the practical applications of PraVFed.

C. Evaluation on Communication and Memory Cost

In this section, we evaluate our method with baseline methods on communication and memory cost under the same experimental configuration. In the experiment, the passive party, which owned a heterogeneous model, assisted the active party in training a global model. We analyze the communication costs required by each method to achieve the target accuracy. Communication costs refer to the total amount of data between active and passive parties. Memory costs refer to the memory used by the model and data during the training process.

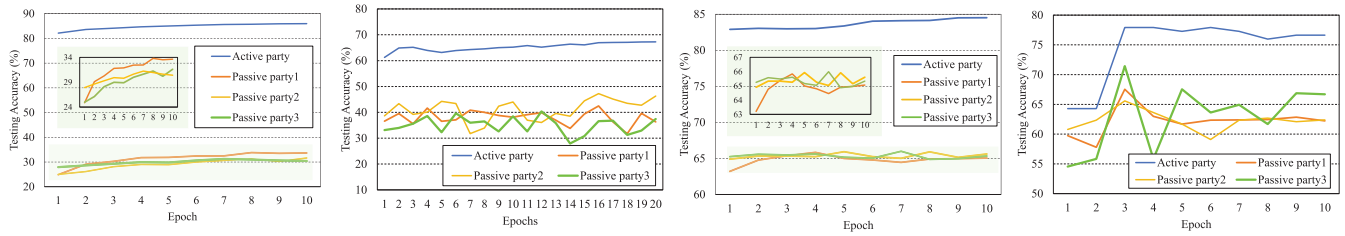
1) Communication Cost: Tables. IV and III plots the communication cost against the testing accuracy on differential datasets. In particular, Table IV presents the convergence accuracy achieved by each method on each image dataset, along with the communication overhead required to reach the target accuracy. Table III presents the convergence accuracy achieved by each method on the tabular dataset, along with the required communication overhead. Table IV indicates that when the target accuracy was set at 85%, Agg_VFL required the least communication cost under the MNIST. This is because all methods require fewer communication rounds to achieve the target accuracy. In the Agg_VFL framework, both active and passive participants exchange solely predictions and gradients during each iteration, which significantly reduces communication overhead. However, the convergence accuracy of Agg_VFL is much lower than that of our method. In contrast, our approach differed in that after the passive party sent its local knowledge to the active party in the first round, the passive party only needed to send its local prediction knowledge to the active party in subsequent global rounds. Moreover, similar observations hold for Table. III. In addition, when the target accuracy is set at 60% under the CINIC dataset, the communication cost of PraVFed is reduced by 70.57% compared to the best baseline Pyvertical method.

2) Memory Cost: During the model training process, the memory requirements of the baseline methods, Pyvertical and Agg_VFL, include the training data and the size of the model. In contrast, the memory requirements of C_VFL consist of the training data, the compressed embeddings of the passive party, and the size of the compressed model. Our method's memory requirements include the training data, the model, and the embedding values from all clients. As a result, our method requires a larger amount of memory compared to the baseline methods. However, the memory requirements of our method remain within an acceptable range for current devices.

Therefore, although our method requires more memory overhead than other baseline methods, its accuracy and communication efficiency significantly outperform those of the other methods.

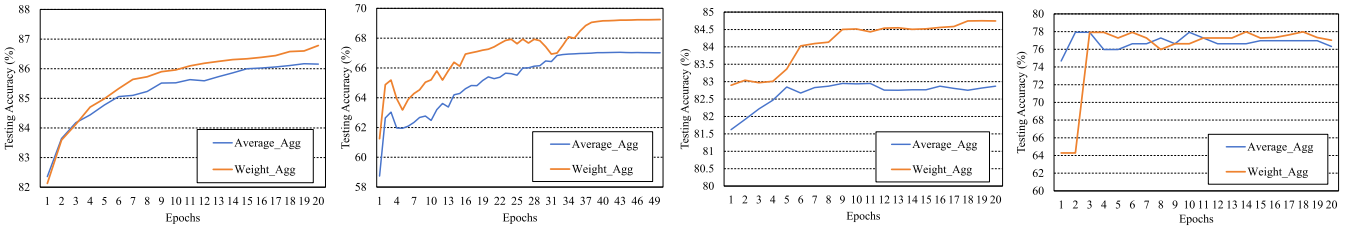
D. Ablation Study

1) Impact of Privacy Budget: In this section, we evaluate the impact of the amount of noise injected into the labels on the global model accuracy. We set the experiment's privacy budgets at 0.1, 0.5, 1.0, 2.0, and 4.0 to evaluate the impact of the active party adding varying degrees of noise to the label on the global model. Tables V and VI present the test accuracy of models jointly developed by clients using heterogeneous models on image and tabular datasets with varying privacy budgets. For example, the client uses the MNIST dataset when $\epsilon = 1.0$, and other passive parties collaborate with the active party to train the θ_1 model employing heterogeneous models, resulting in a model test accuracy of 96.56%. From Table V and VI, we find that the test accuracy of the trained model consistently improves as the privacy budget increases. This is because a larger privacy budget allows for smaller amounts of noise to be injected into the labels, resulting in better performance of the pre-trained model on the passive party.



(a) FMNIST (b) CINIC10 (c) Adult Income (d) Diabetes

Fig. 2. The testing accuracy of the active party's model and passive party's pre-training model under two image datasets and two tabular datasets.



(a) FMNIST (b) CINIC10 (c) Adult Income (d) Diabetes

Fig. 3. The testing accuracy of PraVFed under weight aggregation and average aggregation on two image datasets and two tabular datasets.

TABLE V

AVERAGE TEST ACCURACY OVER FOUR IMAGE DATASETS UNDER HETEROGENEOUS MODEL AND PRIVACY BUDGETS SETTING 0.1, 0.5, 1.0, 2.0, AND 4.0

Datasets	Methods	Testing Accuracy (%)		
		θ_1	θ_2	θ_3
MNIST	$\epsilon = 0.1$	74.67 $\pm$0.15	89.99 $\pm$ 0.04	80.98 $\pm$0.24
	$\epsilon = 0.5$	92.51 $\pm$ 0.06	95.92 $\pm$ 0.06	94.29 $\pm$ 0.04
	$\epsilon = 1.0$	95.35 $\pm$ 0.35	96.56 $\pm$ 0.20	95.56 $\pm$ 0.05
	$\epsilon = 2.0$	95.00 $\pm$ 0.07	97.01 $\pm$ 0.03	95.96 $\pm$ 0.03
	$\epsilon = 4.0$	95.51 $\pm$ 0.02	97.29 $\pm$ 0.02	96.46 $\pm$ 0.04
FMNIST	$\epsilon = 0.1$	81.61 $\pm$ 0.33	83.14 $\pm$ 1.47	84.37 $\pm$ 1.14
	$\epsilon = 0.5$	83.46 $\pm$ 0.11	86.09 $\pm$ 1.10	85.56 $\pm$ 1.50
	$\epsilon = 1.0$	83.76 $\pm$ 0.07	87.41 $\pm$ 1.18	85.12 $\pm$ 0.10
	$\epsilon = 2.0$	83.99 $\pm$ 0.07	87.84 $\pm$ 0.05	86.41 $\pm$ 0.04
	$\epsilon = 4.0$	84.61 $\pm$ 0.13	88.29 $\pm$ 0.05	87.21 $\pm$ 0.06
CIFAR10	$\epsilon = 0.1$	78.75 $\pm$ 0.96	81.25 $\pm$ 0.76	81.61 $\pm$ 0.40
	$\epsilon = 0.5$	81.75 $\pm$ 0.31	82.12 $\pm$ 0.75	81.52 $\pm$ 0.31
	$\epsilon = 1.0$	81.87 $\pm$ 0.25	82.66 $\pm$ 0.13	82.89 $\pm$ 0.30
	$\epsilon = 2.0$	82.87 $\pm$ 1.25	83.38 $\pm$ 0.13	82.89 $\pm$ 0.30
	$\epsilon = 4.0$	82.90 $\pm$ 0.24	83.43 $\pm$ 0.12	83.23 $\pm$ 0.24
CINIC10	$\epsilon = 0.1$	62.01 $\pm$ 0.04	64.42 $\pm$ 0.01	64.16 $\pm$ 0.05
	$\epsilon = 0.5$	64.35 $\pm$ 0.15	64.36 $\pm$ 0.12	64.80 $\pm$ 0.04
	$\epsilon = 1.0$	64.56 $\pm$ 0.03	65.06 $\pm$ 0.13	64.72 $\pm$ 0.04
	$\epsilon = 2.0$	64.79 $\pm$ 0.02	69.23 $\pm$ 0.06	67.47 $\pm$ 0.02
	$\epsilon = 4.0$	65.02 $\pm$ 0.07	69.30 $\pm$ 0.02	65.31 $\pm$ 0.01

TABLE VI

AVERAGE TEST ACCURACY OVER FOUR TABULAR DATASETS AND PRIVACY BUDGETS SETTING 0.1, 0.5, 1.0, 2.0, AND 4.0

	Testing Accuracy (%)			
	Adult Income	Breast Cancer	Credit Card	Diabetes
$\epsilon = 0.1$	84.64 $\pm$ 0.08	96.24 $\pm$ 0.46	99.92 $\pm$ 0.01	75.71 $\pm$ 0.31
$\epsilon = 0.5$	84.71 $\pm$ 0.04	96.49 $\pm$ 0.02	99.93 $\pm$ 0.01	75.87 $\pm$ 0.32
$\epsilon = 1.0$	84.75 $\pm$ 0.10	96.84 $\pm$ 0.20	99.93 $\pm$ 0.01	75.87 $\pm$ 0.32
$\epsilon = 2.0$	84.78 $\pm$ 0.06	96.89 $\pm$ 0.07	99.93 $\pm$ 0.01	76.25 $\pm$ 0.32
$\epsilon = 4.0$	84.78 $\pm$ 0.08	96.91 $\pm$ 0.01	99.94 $\pm$ 0.01	76.28 $\pm$ 0.16

2) Impact of Local Epochs: In this section, we assessed the impact of the number of epochs on both the local model used by the passive party and the global model employed by the active party. We conducted some related evaluation experiments on four datasets. In particular, we get a global model and local heterogeneous models that have already been trained on a set number of local epochs in the experiment. We then evaluated their performance. Figs. 2(a)–2(d) show the results of the model evaluated under different local epoch numbers on the FMNIST, CINIC10, Adult Income, and Diabetes datasets, respectively. It is worth noting that the accuracy of the three local pre-trained models from the passive parties demonstrates a gradual increase as the number of epochs progresses. This suggests that the passive parties are continuously refining their pre-trained models to capture local feature representations, thereby assisting the active party in achieving successful training of the global model. And, when the active party integrates the local embeddings from all passive parties for training, it can access comprehensive feature information. The accuracy of the active party's model is consequently higher and shows a steady upward trend as the number of epochs increases. This indicates that the active party can leverage the feature knowledge from the local models of the passive parties for training, thus enhancing the performance of its model.

3) Impact of Aggregation: In this section, we evaluate the performance of the weight embedding aggregation method. Specifically, we compare the impact of average aggregation (referring to the "Avgerate_Agg" in Fig. 3) and weight aggregation (referring to the "Weight_Agg" in Fig. 3) on model accuracy. In other words, during the training phase, the active party employs averaging and weighted aggregation methods to combine the embeddings from all participants, thereby determining the performance of the main task. To evaluate the performance of weight embedding aggregation across various complex datasets, we conduct comprehensive evaluation experiments utilizing simple image datasets such as FMNIST, as well as complex datasets, including CINIC10,

and two tabular datasets: Adult Income and Diabetes. Each model from the participating entities was self-selected and heterogeneous in nature.

Fig.s 3(a) - 3(d) plot the testing accuracy against the global epochs under both Avgerage_Agg and Weight_Agg. Fig. 3(a) illustrates that the Lenet network can achieve model accuracy of 86.15% and 86.78% on the FMNIST datasets, utilizing Avgerage_Agg and Weight_Agg, respectively. Compared to Avg_Aggre, Weight_Aggre enhances model accuracy by 0.73%. In addition, in the case of the CINIC10, Adult income, and Diabetes datasets, compared to Avgerage_Agg, the Weight_Agg method improved model accuracy by 3.31%, 2.26%, and 0.91%, respectively. Therefore, the model trained using the weight aggregation method achieves higher accuracy compared to the average aggregation method. This limitation arises because the average aggregation method fails to account for the correlation between the feature sets of the data providers and those of the global model. The proposed weight aggregation method quantifies the correlation between local features and models using the accuracy of local pre-trained models. Hence, the weight aggregation method improves the accuracy of the global model.

VI. Conclusion

This paper proposed a novel method called PraVFed to address the challenges of local model heterogeneity and high communication costs. The key idea of PraVFed was to leverage the aggregation of local embedding rather than intermediate results to capture the local knowledge of all participants. To reduce communication costs, we used representation learning to implement passive local multi-round training to obtain embedding values with more local features. We conducted a comprehensive analysis of the effectiveness of PraVFed, considering both theoretical aspects and extensive experimental evaluations. The research results demonstrated that PraVFed significantly reduced communication costs in VFL with heterogeneous models. Our method's practical applicability and advantages contributed to the development and broader adoption of VFL in real-world scenarios.

PraVFed effectively reduces communication costs in VFL. However, PraVFed incurs a significant memory overhead. Future research will investigate the trade-offs between model accuracy, communication overhead, and memory usage, with the goal of adapting VFL to meet the diverse needs of real-world applications.

References

[1] Z. Pan, S. Wang, C. Li, H. Wang, X. Tang, and J. Zhao, "FedMDFG: Federated learning with multi-gradient descent and fair guidance," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, Washington, DC, USA, Jun. 2023, pp. 9364–9371.

[2] Z. Xiong, W. Li, and Z. Cai, "Federated generative model on multi-source heterogeneous data in IoT," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, 2023, pp. 10537–10545.

[3] S. Wang et al., "RAFLS: RDP-based adaptive federated learning with shuffle model," *IEEE Trans. Dependable Secure Comput.*, early access, Jul. 17, 2024, doi: 10.1109/TDSC.2024.3429503.

[4] P. Chatterjee, D. Das, and D. B. Rawat, "Federated learning empowered recommendation model for financial consumer services," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 2508–2516, Feb. 2024.

[5] T. Nevrataki et al., "A survey on federated learning applications in healthcare, finance, and data privacy/data security," *AIP Conf. Proc.*, vol. 2909, Nov. 2023, Art. no. 120015.

[6] D. C. Nguyen et al., "Federated learning for smart healthcare: A survey," *ACM Comput. Surv.*, vol. 55, no. 3, pp. 1–37, Feb. 2022.

[7] J. Huang et al., "Incentive mechanism design of federated learning for recommendation systems in MEC," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 2596–2607, Feb. 2024.

[8] Y. Liu et al., "Vertical federated learning: Concepts, advances and challenges," 2022, *arXiv:2211.12814*.

[9] Q. Zhang, B. Gu, C. Deng, and H. Huang, "Secure bilevel asynchronous vertical federated learning with backward updating," in *Proc. Conf. Artif. Intel.*, 2021, pp. 10896–10904.

[10] S. Wang, K. Gai, J. Yu, and L. Zhu, "BDVFL: Blockchain-based decentralized vertical federated learning," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Shanghai, China, Dec. 2023, pp. 628–637.

[11] W. Xia, Y. Li, L. Zhang, Z. Wu, and X. Yuan, "Cascade vertical federated learning," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2022, pp. 1–6.

[12] Y. Wu et al., "Falcon: A privacy-preserving and interpretable vertical federated learning system," *Proc. VLDB Endowment*, vol. 16, no. 10, pp. 2471–2484, Jun. 2023.

[13] X. Jin, P.-Y. Chen, C.-Y. Hsu, C.-M. Yu, and T. Chen, "CAFE: Catastrophic data leakage in vertical federated learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 34, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 994–1006.

[14] I. Ceballos et al., "SplitNN-driven vertical partitioning," 2020, *arXiv:2008.04137*.

[15] P. Wei et al., "FedAds: A benchmark for privacy-preserving CVR estimation with vertical federated learning," in *Proc. 46th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2023, pp. 3037–3046.

[16] C. Fu et al., "Label inference attacks against vertical federated learning," in *Proc. 31st USENIX Security Symp. (USENIX Secur.)*, 2022, pp. 1397–1414.

[17] K. Wang et al., "FlexiFed: Personalized federated learning for edge clients with heterogeneous model architectures," in *Proc. ACM Web Conf.*, Apr. 2023, pp. 2979–2990.

[18] W. Huang, M. Ye, and B. Du, "Learn from others and be yourself in heterogeneous federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10143–10153.

[19] S. Alam, L. Liu, M. Yan, and M. Zhang, "FedRolex: Model-heterogeneous federated learning with rolling sub-model extraction," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2022, pp. 29677–29690.

[20] J. Zhang, S. Guo, J. Guo, D. Zeng, J. Zhou, and A. Y. Zomaya, "Towards data-independent knowledge transfer in model-heterogeneous federated learning," *IEEE Trans. Comput.*, vol. 72, no. 10, pp. 2888–2901, May 2023.

[21] A. Khan, M. T. Thij, and A. Wilbik, "Communication-efficient vertical federated learning," *Algorithms*, vol. 15, no. 8, p. 273, Aug. 2022.

[22] T. Castiglia, Y. Zhou, S. Wang, S. Kadhe, N. Baracaldo, and S. Patterson, "LESS-VFL: Communication-efficient feature selection for vertical federated learning," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2023, pp. 3757–3781.

[23] J. Wang et al., "TVFL: Tunable vertical federated learning towards communication-efficient model serving," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, New York City, NY, USA, May 2023, pp. 1–10.

[24] F. Fu, X. Miao, J. Jiang, H. Xue, and B. Cui, "Towards communication-efficient vertical federated learning training via cache-enabled local updates," *Proc. VLDB Endowment*, vol. 15, no. 10, pp. 2111–2120, Jun. 2022.

[25] Z. Wu, Q. Li, and B. He, "Practical vertical federated learning with unsupervised representation learning," *IEEE Trans. Big Data*, vol. 10, no. 6, pp. 864–878, Jun. 2024.

[26] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, J. Joshi, and H. Ludwig, "FEDV: Privacy-preserving federated learning over vertically partitioned data," in *Proc. 14th ACM Workshop Artif. Intell. Secur.*, 2021, pp. 181–192.

[27] C.-J. Huang, L. Wang, and X. Han, "Vertical federated knowledge transfer via representation distillation for healthcare collaboration networks," in *Proc. ACM Web Conf.*, Apr. 2023, pp. 4188–4199.

[28] Y. He et al., "Backdoor attack against split neural network-based vertical federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 748–763, 2024.

[29] P. Qiu et al., "Your labels are selling you out: Relation leaks in vertical federated learning," *IEEE Trans. Depend. Secure Comput.*, vol. 20, no. 5, pp. 3653–3668, Sep./Oct. 2023.

[30] L. Hu, H. Yan, L. Li, Z. Pan, X. Liu, and Z. Zhang, "MHAT: An efficient model-heterogenous aggregation training scheme for federated learning," *Inf. Sci.*, vol. 560, pp. 493–503, Jun. 2021.
[31] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12878–12889.
[32] Y. J. Cho, J. Wang, T. Chirvolu, and G. Joshi, "Communication-efficient and model-heterogeneous personalized federated learning via clustered knowledge transfer," *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 1, pp. 234–247, Jan. 2023.
[33] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large CNNs at the edge," in *Proc. NIPS*, vol. 33, 2020, pp. 14068–14080.
[34] T. Castiglia, S. Wang, and S. Patterson, "Flexible vertical federated learning with heterogeneous parties," 2022, *arXiv:2208.12672*.
[35] T. Castiglia, A. Das, S. Wang, and S. Patterson, "Compressed-VFL: Communication-efficient learning with vertically partitioned data," in *Proc. 39th Int. Conf. Mach. Learn.*, 2022, pp. 2738–2766.
[36] Y. Liu et al., "FedBCD: A communication-efficient collaborative learning framework for distributed features," *IEEE Trans. Signal Process.*, vol. 70, pp. 4277–4290, 2022.
[37] M. Abadi et al., "Deep learning with differential privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 308–318.
[38] Y. Tan et al., "FedProto: Federated prototype learning across heterogeneous clients," in *Proc. 36th AAAI Conf. Artif. Intell.*, 2022, pp. 8432–8440.
[39] Y. Tan, G. Long, J. Ma, L. Liu, T. Zhou, and J. Jiang, "Federated learning from pre-trained models: A contrastive learning approach," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2022, pp. 19332–19344.
[40] P. Qiu, X. Zhang, S. Ji, C. Fu, X. Yang, and T. Wang, "HashVFL: Defending against data reconstruction attacks in vertical federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 3435–3450, 2024.
[41] T. Zou, Z. Gu, Y. He, H. Takahashi, Y. Liu, and Y.-Q. Zhang, "VFLAIR: A research library and benchmark for vertical federated learning," 2023, *arXiv:2310.09827*.
[42] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[43] X. Han, R. Kashif, and V. Roland, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
[44] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features From Tiny Images*. Toronto, ON, Canada: Virtual Event, 2009.
[45] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-IID data," in *Proc. NIPS*, 2021, pp. 5972–5984.
[46] F. Ding, M. Hardt, J. A. Miller, and L. Schmidt, "Retiring adult: New datasets for fair machine learning," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2021, pp. 6478–6490.
[47] W. N. Street, W. H. Wolberg, and O. L. Mangasarian, "Nuclear feature extraction for breast tumor diagnosis," *Proc. SPIE*, vol. 1905, pp. 861–870, Jul. 1993.
[48] W. Zheng, L. Yan, C. Gou, and F.-Y. Wang, "Federated meta-learning for fraudulent credit card detection," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 4654–4660.
[49] U. M. Butt, S. Letchmunan, M. Ali, F. H. Hassan, A. Baqir, and H. H. R. Sherazi, "Machine learning based diabetes classification and prediction for healthcare applications," *J. Healthcare Eng.*, vol. 2021, pp. 1–17, Sep. 2021.
[50] X. Yuan, W. Ni, M. Ding, K. Wei, J. Li, and H. V. Poor, "Amplitude-varying perturbation for balancing privacy and utility in federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1884–1897, 2023.
[51] D. Romanini et al., "PyVertical: A vertical federated learning framework for multi-headed SplitNN," 2021, *arXiv:2104.00489*.
[52] J. Zhang et al., "Adaptive vertical federated learning on unbalanced features," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4006–4018, Dec. 2022.
[53] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NIPS*, Dec. 2019, pp. 8024–8035.

Shuo Wang (Student Member, IEEE) is currently pursuing the Ph.D. degree in electronic information with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. Her current research interests include federated learning, privacy-preserving computation, and blockchain.

Keke Gai (Senior Member, IEEE) received the Ph.D. degree in computer science from Pace University, New York, NY, USA. He is currently a Professor at the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. He has published five books and more than 180 peer-reviewed journals. His research interests include cybersecurity, blockchain, privacy-preserving computation, and decentralized identity. He serves as the Co-Chair for IEEE Technology and Engineering Management Society's Technical Committee on Blockchain and Distributed Ledger Technologies and a Standing Committee Member of China Computer Federation-Blockchain Committee. He also serves as the Editor-in-Chief for the journal *Blockchains* and an Associate Editor for a few journals, including IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING and *Journal of Parallel and Distributed Computing*.

Jing Yu (Member, IEEE) received the B.S. degree in automation science from the Minzu University of China, Beijing, China, in 2011, the M.S. degree in pattern recognition from Beihang University, China, in 2014, and the Ph.D. degree from the University of Chinese Academy of Sciences, China, in 2019. She is currently an Associate Professor with the School of Information Engineering, Minzu University of China. Her research interests include cross-modal understanding and artificial intelligence security.

Zijian Zhang (Senior Member, IEEE) is currently a Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology, China. He has published more than 100 research papers in international peer-reviewed conferences, transactions, and magazines, including ICML, WWW, AAAI, INFOCOM, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and IEEE TRANSACTIONS ON MOBILE COMPUTING. His current research interests include anonymous communication, blockchain traffic analysis, consensus, and vulnerability detection of smart contracts. He won the Best Paper Award from IWQoS'17, ICC'24, and SmartCom'22.

Liehuang Zhu (Senior Member, IEEE) is currently a Full Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology. He is selected into the Program for New Century Excellent Talents in University from the Ministry of Education, China. He has published over 60 SCI-indexed research articles in these areas, and a book published by Springer. His research interests include the Internet of Things, cloud computing security, internet, and mobile security. He serves on the editorial boards of three international journals, including IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and *IEEE Network* magazine. He won the Best Paper Award at IEEE/ACM IWQoS 2017 and IEEE TrustCom 2018.