FedVS: Straggler-Resilient and Privacy-Preserving Vertical Federated Learning for Split Models

Songze Li [1] [2] **Duanyi Yao** [2] **Jin Liu** [1]

Abstract

In a vertical federated learning (VFL) system consisting of a central server and many distributed clients, the training data are vertically partitioned such that different features are privately stored on different clients. The problem of split VFL is to train a model split between the server and the clients. This paper aims to address two major challenges in split VFL: 1) performance degradation due to straggling clients during training; and 2) data and model privacy leakage from clients' uploaded data embeddings. We propose FedVS to simultaneously address these two challenges. The key idea of FedVS is to design secret sharing schemes for the local data and models, such that information-theoretical privacy against colluding clients and curious server is guaranteed, and the aggregation of *all* clients' embeddings is reconstructed losslessly, via decrypting computation shares from the non-straggling clients. Extensive experiments on various types of VFL datasets (including tabular, CV, and multi-view) demonstrate the universal advantages of FedVS in straggler mitigation and privacy protection over baseline protocols.

1. Introduction

Federated learning (FL) (McMahan et al., 2017; Zhang et al., 2021a) is an emerging machine learning paradigm where multiple clients (e.g., companies) collaborate to train a machine learning model while keeping the raw data decentralized. Based on how data is partitioned across clients, FL can be categorized into horizontal FL and vertical FL. In horizontal FL (HFL), each client possesses a distinct set of data samples who share the same set of features; in vertical FL (VFL), each client has a distinct subset of features for a collection of shared samples. While current FL research largely focused on HFL, VFL is attracting more attention due to its suitability for enabling data augmentation for a wide range of applications in decision making (Cheng et al., 2021), risk control (Cheng et al., 2020), and health care (Lee et al., 2018). In a basic VFL setting (see, e.g., (Yang et al., 2019a; Feng & Yu, 2020)), the FL system trains a local model for each client, which are jointly utilized to perform inferences. A more general VFL setting, named split VFL (Ceballos et al., 2020), incorporates the idea of split learning (Vepakomma et al., 2018), and jointly trains a central model at the server and local models at the clients.

In a training round of split VFL, all clients forward propagate their local data using local models, and send the output embeddings to the server; the server then aggregates these embeddings and continues forward prorogation through its central model. Having computed the loss, the server back propagates to update the central model, and sends the gradients of the embeddings to the clients to update the local models. An ideal round requires synchronous aggregation of clients' embeddings. However, this is severely challenged by the system and task heterogeneity commonly observed in VFL, which is caused by variability of clients' storage, computation and communication resources, and local data and model complexities (Reisizadeh et al., 2022; Wei et al., 2022). Clients with slowest speeds of forward propagation, which we call *stragglers*, become the bottleneck in training process, and cause detrimental effects on model convergence.

One way to deal with stragglers is simply ignoring them, which however leads to slow convergence and model bias. Asynchronous VFL protocols have been proposed to enable asynchronous submissions of embeddings and model updates without client coordination (Chen et al., 2020; Hu et al., 2019). However, this causes staleness of model updates that can degrade model performance. Under the synchronous framework, Flex-VFL (Castiglia et al., 2022) was proposed to enable flexible numbers of local model updates across clients, mitigating the slowdown of convergence caused by stragglers.

Other than stragglers, another key challenge for split VFL

[1] The Hong Kong University of Science and Technology (Guangzhou) [2] The Hong Kong University of Science and Technology. Correspondence to: Songze Li <songzeli@ust.hk>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

is privacy leakage through clients' embeddings. Various inference attacks have been developed to recover clients' private data and model parameters, from the uploaded raw embeddings ((Erdogan et al., 2021; Jin et al., 2021; Li et al., 2021a; Luo et al., 2021; Fu et al., 2022)). Differential privacy (DP) has been adopted to defend inference attacks, which adds a DP noise layer on raw embeddings to protect data privacy (see, e.g., (Thapa et al., 2022; Chen et al., 2020; Xu et al., 2021)). However, the added noises cause inaccurate computations of gradients, which subsequently leads to performance loss. Homomorphic encryption (HE) has also been utilized in VFL to protect embedding privacy, such that ciphertexts of embeddings are aggregated and only the summation of all embeddings is revealed (Hardy et al., 2017; Yang et al., 2019b; Cai et al., 2022). These methods provide privacy for clients' data but cannot mitigate stragglers effectively. Recently in (Shi et al., 2022), it is proposed to use secure aggregation (Bonawitz et al., 2017) for privacy protection in asynchronous training of linear and logistic regression models over vertically partitioned data, which is nevertheless faced with slow convergence from asynchronous model updates. Given the above challenges and the prior works, we ask the following question:

Can one design a synchronous split VFL protocol that is simultaneously lossless against unknown stragglers and provably private against curious server and clients?

We answer this question in affirmative, via proposing a straggler-resilient and privacy-preserving split VFL protocol named FedVS. The key idea is to secret share local data and model of each client with peer clients, creating data redundancy across the network without any privacy leakage. Specifically, Lagrange Coded Computing (LCC) (Yu et al., 2019) is adopted to improve computation and communication efficiencies. Averaging is chosen as the embedding aggregation method, such that the server only recovers the summation of the embeddings without knowing individual values. Clients utilize polynomial networks (Livni et al., 2014) as local models, such that embedding summation can be *losslessly* reconstructed at the server using polynomial interpolation. Leveraging the threshold property of polynomial interpolation, computation results from only a subset of clients are needed, effectively mitigating the stragglers. We theoretically analyze the straggler resilience and privacy guarantees of FedVS, its convergence performance, and operational complexities.

We experimentally demonstrate the advantages of FedVS in straggler mitigation and privacy protection for split VFL systems. Over a wide range of tabular, computer vision, and multi-view datasets, FedVS uniformly achieves the fastest convergence and highest accuracy, over baselines with or without privacy protection. The impacts of design parameters of FedVS on its performance and privacy are also empirically studied.

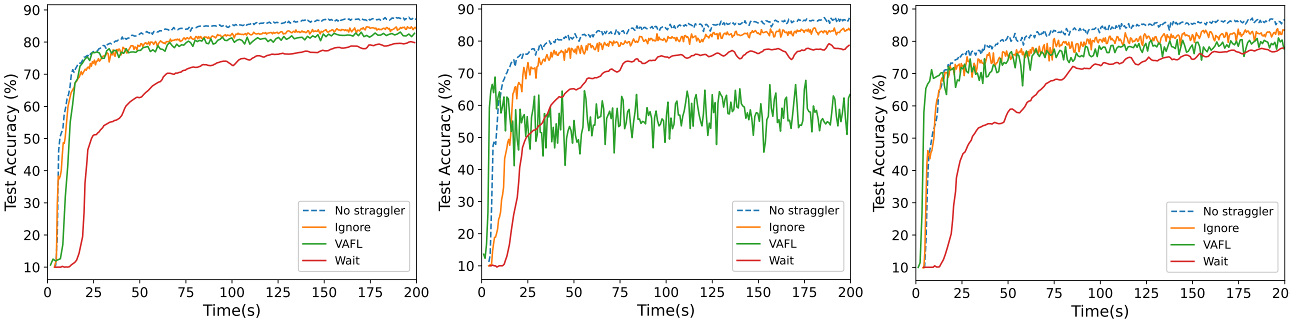
Related works

Straggler-resilient FL:

Horizontal FL: Proposed in (Reisizadeh et al., 2022), FLANP starts the training with server exchanging models with a group of fast-responding clients, and gradually involves the slower clients. Sageflow (Park et al., 2021) proposes to group the local models from stragglers according to their staleness, and aggregate the models from different groups with appropriate weights. In (Dhakal et al., 2019; Prakash et al., 2020; Sun et al., 2022a;b), clients share a part of their local data with the server, who computes the missing results from stragglers; while in (Schlegel et al., 2021; Shao et al., 2022), clients secret share their data with each other and perform local training on shares of all clients, such that the server losslessly decodes the gradient over all clients' data from only a subset of non-straggling clients. On the other hand, many asynchronous HFL protocols (Xie et al., 2019; van Dijk et al., 2020; Li et al., 2021b; Huba et al., 2022; Chai et al., 2021; Nguyen et al., 2022) have been proposed to handle the straggler problem.

Vertical FL: For mitigating stragglers in VFL, Multiple asynchronous VFL protocols (see, e.g., (Chen et al., 2020; Gu et al., 2021; Zhang et al., 2021b; Li et al., 2020; Shi et al., 2022; Hu et al., 2019)) have been proposed to reduce the waiting time for stragglers. VAFL (Chen et al., 2020) is designed for clients with intermittent connectivities, where each client individually updates its local model once connected with the server. DP is introduced to protect the privacy of local embeddings in VAFL, which nevertheless incurs performance loss. AFSGD-VP (Gu et al., 2021) is designed for the scenario where there is no central server and labels are held by multiple clients. It allows asynchronous data collection and model updating for label holders, and at the same time protects embedding privacy via a tree-structured aggregation scheme. AMVFL (Shi et al., 2022) proposes asynchronous aggregation to compute gradients, for linear and logistic regression problems, where local embeddings are protected by secret shared masks.

Privacy-preserving FL: Current approaches to provide privacy protection for FL can be categorized into three types, which are homomorphic encryption (HE), DP, and secure multi-party computation (MPC) (Liu et al., 2022b). HE methods are applied to encrypt the local updates sent to the server (see, e.g.,(Chai et al., 2020; Zhang et al., 2020; Cai et al., 2022)). It allows certain computations (e.g., addition) directly on the ciphertexts and noise-free recovery of computation results. However, the encryption and decryption introduce significant overheads. Compared with HE, DP is more efficient to provide privacy by injecting noises to the private data (Wei et al., 2020; Truex et al.,



(a) Concatenation

(b) Element-wise average

(c) Element-wise maximum

Figure 1. Test accuracies using different embedding aggregation methods and straggler handling strategies.

2020; Thapa et al., 2022; Wang et al., 2020). Nevertheless, the performance and convergence rate of the model suffer from the inaccurate computation results (Truex et al., 2019; Kairouz et al., 2021). MPC protocols based on Shamir secret sharing have been proposed to securely aggregate clients' local models in HFL, such that the server learns nothing beyond the aggregated model (Bonawitz et al., 2017; So et al., 2021; Bell et al., 2020; Choi et al., 2020; So et al., 2022; Liu et al., 2022a; Jahani-Nezhad et al., 2022a;b). These protocols guarantee information-theoretic privacy for clients' local data, in the presence of client dropouts. Compared with these works, the proposed FedVS is the first MPC-based synchronous VFL protocol that simultaneously achieves information-theoretic privacy for each client's local data and model. Furthermore, in contrast to recovering model aggregation of non-straggling clients, FedVS achieves straggler resilience with no performance loss, i.e., the recovered embedding aggregation contains the local embeddings of all stragglers.

2. Background and Motivations

2.1. Split vertical federated learning

We consider a vertical federated learning (VFL) system that consists of a central server and N clients. The training dataset $\mathcal{S} = \{(\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)})\}_{m=1}^{M}$ contains M input-label pairs, where each input $\boldsymbol{x}^{(m)} \in \mathbb{R}^d$ has d features. The training set is vertically partitioned such that each client n locally has a disjoint subset of d_n features of each input. All labels are stored at the server. The VFL system aims to train a neural network that is split among server and clients. The server has a central model with parameters $\boldsymbol{W}_0$, and each client n has a local model with parameters $\boldsymbol{W}_n$. Models on different clients may have different architectures, and hence the model parameters may have different dimensions.

The server and clients collaboratively train their models to minimize the empirical loss $L((\boldsymbol{W}_n)_{n=0}^{N}; \mathcal{S}) = \frac{1}{M}\sum_{m=1}^{M} \ell\left((\boldsymbol{W}_n)_{n=0}^{N}; (\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)})\right)$, for some loss function ℓ. The training is carried out via forward-backward prorogation over split models. In each round, for a batch $\mathcal{B}$ of b inputs $\boldsymbol{X}^{(\mathcal{B})} \in \mathbb{R}^{b\times d}$, we denote the partition at client n as $\boldsymbol{X}_n^{(\mathcal{B})} \in \mathbb{R}^{b\times d_n}$, for all $n \in [N] \triangleq \{1, \ldots, N\}$. To start, each client n computes an embedding matrix $\boldsymbol{H}_n^{(\mathcal{B})} \in \mathbb{R}^{b\times h_n}$, for some embedding dimension h_n, using its local network as $\boldsymbol{H}_n^{(\mathcal{B})} = g_n(\boldsymbol{X}_n^{(\mathcal{B})}, \boldsymbol{W}_n)$, and sends it to the server. The server aggregates embeddings from all clients into a global embedding $\boldsymbol{H}^{(\mathcal{B})}$. As discussed in (Ceballos et al., 2020), the aggregation can be done in multiple ways, including concatenation, element-wise average, and element-wise maximum. Next, the server feeds $\boldsymbol{H}^{(\mathcal{B})}$ into the central network until the loss function L is computed with the corresponding labels $\boldsymbol{Y}^{(\mathcal{B})}$. In the backward propagation, the server computes the gradient $\nabla_{\boldsymbol{W}_0} L$ to update the central model with learning rate η_0, i.e., $\boldsymbol{W}_0 = \boldsymbol{W}_0 - \eta_0 \nabla_{\boldsymbol{W}_0} L$. Then, for each $n \in [N]$, the server computes the gradient $\nabla_{\boldsymbol{H}_n^{(\mathcal{B})}} L$ and sends it to client n. Finally, each client n further computes the gradient with respect to its local model, and updates the local model with learning rate η_n, i.e., $\boldsymbol{W}_n = \boldsymbol{W}_n - \eta_n \nabla_{\boldsymbol{H}_n^{(\mathcal{B})}} L \cdot \nabla_{\boldsymbol{W}_n} \boldsymbol{H}_n^{(\mathcal{B})}$.

2.2. Straggler and privacy challenges

Challenge 1: Performance degradation from stragglers. Straggler problem is commonly observed in FL systems, due to heterogeneous computation and communication resources across clients, and can be even worse for VFL systems where heterogeneity also exists for local model architecture and data features. To understand the effect of stragglers on model performance, we carry out experiments on the FashionMNIST dataset (Xiao et al., 2017a) in split VFL setting, where 16 clients evenly hold parts of each training image. We select 60% clients as stragglers to add an additional exponential delay when submitting their embeddings. We compare three strategies to handle stragglers: 1) Wait for all stragglers (Wait); 2) Ignore stragglers (Ignore); and 3) VAFL with asynchronous model updates (Chen et al., 2020). Three methods, including concatenation, element-wise average and element-wise maximum, are utilized for embedding aggregation. As shown in Figure 1, for all aggregation methods and strategies, presence of stragglers leads to convergence slowdown and accuracy degradation.

Challenge 2: Data/model leakage. The embedding from a client contains information about its private data and local model parameters. It has been shown in (Luo et al., 2021; Erdogan et al., 2021) that through inference attacks, a curious server can reconstruct a victim client's private input features and local model, from its uploaded embedding.

Threat model. We consider an *honest-but-curious* threat model, which is widely adopted to study the privacy vulnerabilities of FL systems. All parties in the system will faithfully follow the specified learning protocol. The curious server attempts to infer private data and local model of a victim client from its uploaded computation results. A subset of curious clients may collude to infer the private data and local models of the other victim clients.

The goal of this work is to tackle the above challenges, via developing a synchronous split VFL framework whose model training is resilient to stragglers, and private against passively inferring clients' local data and model parameters.

3. Preliminaries

Embedding averaging. We adopt the element-wise average as the aggregation method. That is, $\boldsymbol{H}^{(\mathcal{B})} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{H}_n^{(\mathcal{B})}$. The reason for this choice is two-folded: 1) As shown in Figure 1, compared with concatenation, which is the best performing aggregation method, element-wise average achieves comparable performance when there is no straggler; 2) For element-wise average, the server does not necessarily need to know *individual* client embeddings to compute their summation, hence potentially allowing a higher level of privacy protection. To implement this embedding averaging, we require the same dimension for the embeddings from all clients, i.e., $h_1 = \cdots = h_N = h$.

Lagrange coded computing. Proposed in (Yu et al., 2019), Lagrange coded computing (LCC) is a cryptographic primitive for sharing multiple secrets. Given a privacy parameter T, LCC guarantees information-theoretic privacy against up to T colluding shares. LCC supports homomorphic evaluation of arbitrary polynomials on the shares. The decryption is accomplished through polynomial interpolation, which is resilient to loss of decryption shares up to a certain threshold.

Polynomial networks. As one of our main goals is to provide data and model privacy for split VFL, which requires utilizing secure computation primitives like LCC, we adopt polynomial network (PN) as the architecture of the client models. Proposed in (Livni et al., 2014), a PN uses quadratic function as the activation function, and outputs a polynomial function of the input. For instance, the output $y \in \mathbb{R}$ of a 2-layer PN with r neurons in the hidden layer, for some input $\boldsymbol{x} \in \mathbb{R}^d$, is computed as $y = b + \boldsymbol{w}_0^\top \boldsymbol{x} + \sum_{i=1}^{r} \alpha_i (\boldsymbol{w}_i^\top \boldsymbol{x})^2$, where $\boldsymbol{w}_i \in \mathbb{R}^d$ are network parameters. Compared with standard architectures like MLP and CNN with non-linear activation functions, PN is natively compatible with homomorphic evaluations on secret shares, and at the same time exhibited superior performance (Liu et al., 2021). Here we consider a simplified architecture such that for a PN with D layers, the output embedding $\boldsymbol{h} \in \mathbb{R}^h$ is produced from an input $\boldsymbol{x} \in \mathbb{R}^d$ as $\boldsymbol{h} = \sum_{i=1}^{D}(\boldsymbol{x}^i \boldsymbol{W}^i + \boldsymbol{b}^i)$, where $\boldsymbol{x}^i$ is the ith power of the input computed element-wise, and $\boldsymbol{W}^i \in \mathbb{R}^{d \times h}$ and $\boldsymbol{b}^i \in \mathbb{R}^h$ are the weight matrix and bias vector for the ith layer.

Table 1. Test accuracies of different client network architectures.

# of layers	MLP	CNN	PN
1	88.35%	90.48%	88.19%
2	88.60%	90.79%	88.31%
3	88.70%	91.53%	88.49%

In a split VFL system, a PN with D_n layers at client n consists of D_n weight matrices $\boldsymbol{W}_n = (\boldsymbol{W}_n^1, \ldots, \boldsymbol{W}_n^{D_n})$, where $\boldsymbol{W}_n^i \in \mathbb{R}^{d_n \times h}$. For an input data partition $\boldsymbol{X}_n^{(\mathcal{B})}$ of batch $\mathcal{B}$, the output embeddings are computed as

$$\boldsymbol{H}_n^{(\mathcal{B})} = g_n(\boldsymbol{X}_n^{(\mathcal{B})}, \boldsymbol{W}_n) = \sum_{i=1}^{D_n} \boldsymbol{X}_n^{i,(\mathcal{B})} \boldsymbol{W}_n^i, \text{[1]} \tag{1}$$

where $\boldsymbol{X}_n^{i,(\mathcal{B})}$ is a matrix whose elements are ith power of the corresponding elements in $\boldsymbol{X}_n^{(\mathcal{B})}$.

To verify the effectiveness of using PN in split VFL, we train image classifiers on FashionMNIST over 4 clients. The server holds a VGG13 network (Simonyan & Zisserman, 2014); three different network architectures, including MLP, CNN, and PN, are respectively employed at the clients. As shown in Table 1, PN achieves comparable performance with CNN. which has the highest accuracies.

4. Protocol Description

4.1. Overview

We develop a synchronous split VFL framework FedVS, which simultaneously addresses the straggler and privacy leakage challenges. In FedVS, each client secret shares its training data across the network using LCC before training starts. In each training round, each client first secret shares its current local model; then, utilizing the algebraic structures of the shares and the underlying PN computation, each client performs homomorphic evaluations on coded data and models, and sends computation results to the server. The summation of embeddings can be reconstructed losslessly at the server, in spite of missing results from a

[1] The bias vectors are absorbed into the weight matrices with a 1 appended to each data sample.

threshold number of stragglers. We give a full description of FedVS in Algorithm 1.

4.2. Data preparation

Before training starts, a data preparation step takes place among the clients.

Pre-processing and quantization. Each client n pre-processes its input $\boldsymbol{X}_n$ to obtain $\widehat{\boldsymbol{X}}_n = (\boldsymbol{X}_n^1, \ldots, \boldsymbol{X}_n^{D_n})$, where $\boldsymbol{X}_n^i$ is computed via raising $\boldsymbol{X}_n$ to the ith power element-wise. Then, the client quantizes $\widehat{\boldsymbol{X}}_n$ onto a finite field $\mathbb{F}_p$, for some sufficiently large prime p. Specifically, for some scaling factor l_x, rounding operator $\text{Round}(x) = \begin{cases} \lfloor x \rfloor, & \text{if } x - \lfloor x \rfloor < 0.5 \\ \lfloor x \rfloor + 1, & \text{otherwise} \end{cases}$, and shift operator $\phi(x) = \begin{cases} x, & \text{if } x \geq 0 \\ p + x, & \text{if } x < 0 \end{cases}$, client n obtains its quantized data $\overline{\boldsymbol{X}}_n = \phi(\text{Round}(2^{l_x} \cdot \widehat{\boldsymbol{X}}_n))$, applied element-wise.

Private data sharing. The clients secret share their quantized local data with other clients using LCC with *partition parameter* K and *privacy parameter* T. Specifically, for each $n \in [N]$, client n horizontally partitions $\overline{\boldsymbol{X}}_n = (\overline{\boldsymbol{X}}_n^1, \ldots, \overline{\boldsymbol{X}}_n^{D_n})$ into K segments $\overline{\boldsymbol{X}}_{n,1}, \ldots, \overline{\boldsymbol{X}}_{n,K}$, and then samples independently T masks $\boldsymbol{Z}_{n,K+1}, \ldots, \boldsymbol{Z}_{n,K+T}$ uniformly at random. For a set of distinct parameters $\{\beta_1, \ldots, \beta_{K+T}\}$ from $\mathbb{F}_p$ that are agreed among all clients and the server, using Lagrange interpolation, client n obtains the following polynomial.

$$\begin{aligned} \boldsymbol{F}_n(x) = &\sum_{k=1}^{K} \overline{\boldsymbol{X}}_{n,k} \cdot \prod_{\ell \in [K+T] \setminus \{k\}} \frac{x - \beta_\ell}{\beta_k - \beta_\ell} \\ &+ \sum_{k=K+1}^{K+T} \boldsymbol{Z}_{n,k} \cdot \prod_{\ell \in [K+T] \setminus \{k\}} \frac{x - \beta_\ell}{\beta_k - \beta_\ell}. \end{aligned} \tag{2}$$

Here we note that $F_n(\beta_k) = \overline{\boldsymbol{X}}_{n,k}$, for all $k \in [K]$.

For another set of public parameters $\{\alpha_1, \ldots, \alpha_N\}$ that are pair-wise distinct and $\{\beta_1, \ldots, \beta_{K+T}\} \cap \{\alpha_1, \ldots, \alpha_N\} = \varnothing$, client n computes $\widetilde{\boldsymbol{X}}_{n,n'} = \boldsymbol{F}_n(\alpha_{n'})$, for all $n' \in [N]$, and sends it to client n'. Note that the size of a secret share is $\frac{1}{K}$ of the size of the original data. Data partitioning in LCC helps to reduce the communication cost for secret sharing, and the complexity of subsequent computations on secret shares. By the end of the data sharing phase, each client n' has locally the secret shares $\widetilde{\boldsymbol{X}}_{n'} = (\widetilde{\boldsymbol{X}}_{1,n'}, \ldots, \widetilde{\boldsymbol{X}}_{N,n'})$ from all N clients.

4.3. Training operations

Model quantization and secret sharing. A training round starts with each client n quantizing and secret sharing its

Algorithm 1 The FedVS protocol

Input: K (partition parameter), T (privacy parameter)

```
// Data preparation phase
for each client $n = 1, 2, \ldots, N$ in parallel do
    $\widehat{\boldsymbol{X}}_n \leftarrow (\boldsymbol{X}_n^1, \ldots, \boldsymbol{X}_n^{D_n})$ // Raises data to the degree of local PN
    $\overline{\boldsymbol{X}}_n \leftarrow$ Quantization on $\widehat{\boldsymbol{X}}_n$
    $\overline{\boldsymbol{X}}_{n,1}, \ldots, \overline{\boldsymbol{X}}_{n,K} \leftarrow$ Horizontally partitions $\overline{\boldsymbol{X}}_n$ into $K$ segments
    $\boldsymbol{Z}_{n,K+1}, \ldots, \boldsymbol{Z}_{n,K+T} \leftarrow$ Sample random masks
    $\{\widetilde{\boldsymbol{X}}_{n,n'}\}_{n' \in [N]} \leftarrow$ Evaluating (2) at $\alpha_1, \ldots, \alpha_N$ // Data secret shares
    Sends data share $\widetilde{\boldsymbol{X}}_{n,n'}$ to client $n' \in [N] \backslash \{n\}$
    Receives data share $\widetilde{\boldsymbol{X}}_{n',n}$ from user $n' \in [N] \backslash \{n\}$
end for

// Training phase
for Round $1, 2, \ldots$ do
    // Model secret sharing
    for each client $n = 1, 2, \ldots, N$ in parallel do
        $\overline{\boldsymbol{W}}_n \leftarrow$ Quantization on $\boldsymbol{W}_n$
        $\boldsymbol{V}_{n,K+1}, \ldots, \boldsymbol{V}_{n,K+T} \leftarrow$ Sample random masks
        $\{\widetilde{\boldsymbol{W}}_{n,n'}\}_{n' \in [N]} \leftarrow$ Evaluating (4) at $\alpha_1, \ldots, \alpha_N$ // Model secret shares
        Sends model share $\widetilde{\boldsymbol{W}}_{n,n'}$ to client $n' \in [N] \backslash \{n\}$
        Receives model share $\widetilde{\boldsymbol{W}}_{n',n}$ from user $n' \in [N] \backslash \{n\}$
    end for
    // Homomorphic embedding evaluation
    for each client $n = 1, 2, \ldots, N$ in parallel do
        For a sample batch $\mathcal{B}$, computes coded embedding $\widetilde{\boldsymbol{H}}_n^{(\mathcal{B})}$ as in (5) and sends it to server
    end for
    // Server model update
    Server executes:
    Receives coded embeddings from non-straggling clients $\mathcal{U} \subset [N]$
    Interpolates embedding summation polynomial $\psi(x)$ in (6) from $\{\widetilde{\boldsymbol{H}}_n^{(\mathcal{B})} : n \in \mathcal{U}\}$
    Recovers embedding summation $\overline{\boldsymbol{H}}^{(\mathcal{B})}$ by evaluating $\psi(x)$ at $\beta_1, \ldots, \beta_K$
    $\boldsymbol{H}^{(\mathcal{B})} \leftarrow$ Dequantization on $\overline{\boldsymbol{H}}^{(\mathcal{B})}$ // Recovers average embedding over all clients (including stragglers)
    Back-propogates to update central model $\boldsymbol{W}_0$, and broadcasts $\nabla_{\boldsymbol{H}^{(\mathcal{B})}} L$ to all clients
    // Client model update
    for each client $n = 1, 2, \ldots, N$ in parallel do
        Obtains $\nabla_{\boldsymbol{W}_n} L \leftarrow \nabla_{\boldsymbol{H}_n^{(\mathcal{B})}} L \cdot \nabla_{\boldsymbol{W}_n} \boldsymbol{H}_n^{(\mathcal{B})}$, and updates local model $\boldsymbol{W}_n$
    end for
end for
```

current model parameters $\boldsymbol{W}_n$. Firstly, For some scaling factor l_w, client n quantizes its model parameters to obtain

$$\overline{\boldsymbol{W}}_n=(\overline{\boldsymbol{W}}_n^1,\dots,\overline{\boldsymbol{W}}_n^{D_n})=\phi(\text{Round}_{stoc}(2^{l_w}\cdot\boldsymbol{W}_n)). \quad (3)$$

Here $\text{Round}_{stoc}(x) = \begin{cases}\lfloor x\rfloor & \text{with prob. } 1-(x-\lfloor x\rfloor)\\ \lfloor x\rfloor+1 & \text{with prob. } x-\lfloor x\rfloor\end{cases}$ is an unbiased stochastic rounding operator, i.e., $\mathbb{E}[\text{Round}_{stoc}(x)]=x$.

Then, client n samples uniformly at random T noise terms $\boldsymbol{V}_{n,K+1},\dots,\boldsymbol{V}_{n,K+T}$, and constructs the following Lagrange polynomial.

$$\begin{aligned}\boldsymbol{G}_n(x)=&\sum_{k=1}^{K}\overline{\boldsymbol{W}}_n\cdot\prod_{\ell\in[K+T]\setminus\{k\}}\frac{x-\beta_\ell}{\beta_k-\beta_\ell}\\&+\sum_{k=K+1}^{K+T}\boldsymbol{V}_{n,k}\cdot\prod_{\ell\in[K+T]\setminus\{k\}}\frac{x-\beta_\ell}{\beta_k-\beta_\ell}.\end{aligned}\quad (4)$$

For each $n'\in[N]$, client n computes a secret share of it model $\widetilde{\boldsymbol{W}}_{n,n'}=\boldsymbol{G}_n(\alpha_{n'})$, and sends it to client n'.[2]

Homormophic evaluation and embedding decryption. For a batch $\mathcal{B}\subseteq[\frac{M}{K}]$ of coded training samples, the clients start forward propagation by homomorphic embedding evaluation. Specifically, for each $n'\in[N]$, client n' takes the coded data $\widetilde{\boldsymbol{X}}^{(\mathcal{B})}_{1,n'},\dots,\widetilde{\boldsymbol{X}}^{(\mathcal{B})}_{N,n'}$, with $\widetilde{\boldsymbol{X}}^{(\mathcal{B})}_{n,n'}=(\widetilde{\boldsymbol{X}}^{1,(\mathcal{B})}_{n,n'},\dots,\widetilde{\boldsymbol{X}}^{D_n,(\mathcal{B})}_{n,n'})$ for all $n\in[N]$, and the coded models $\widetilde{\boldsymbol{W}}_{1,n'},\dots,\widetilde{\boldsymbol{W}}_{N,n'}$, with $\widetilde{\boldsymbol{W}}_{n,n'}=(\widetilde{\boldsymbol{W}}^1_{n,n'}\dots,\widetilde{\boldsymbol{W}}^{D_n}_{n,n'})$ for all $n\in[N]$, computes its output

$$\widetilde{\boldsymbol{H}}^{(\mathcal{B})}_{n'}=\sum_{n=1}^{N}g_n(\widetilde{\boldsymbol{X}}^{(\mathcal{B})}_{n,n'},\widetilde{\boldsymbol{W}}_{n,n'})=\sum_{n=1}^{N}\sum_{i=1}^{D_n}\widetilde{\boldsymbol{X}}^{i,(\mathcal{B})}_{n,n'}\widetilde{\boldsymbol{W}}^i_{n,n'}, \quad (5)$$

and sends $\widetilde{\boldsymbol{H}}^{(\mathcal{B})}_{n'}$ to the server. During this process, some clients become stragglers, and server only waits to receive results from a subset $\mathcal{U}\subset[N]$ of non-straggling clients.

It is easy to see that for the polynomial $\boldsymbol{F}^{(\mathcal{B})}_n(x)=(\boldsymbol{F}^{1,(\mathcal{B})}_n(x)\dots,\boldsymbol{F}^{D_n,(\mathcal{B})}_n(x))$ corresponding to data batch $\mathcal{B}$, and the model polynomial $\boldsymbol{G}_n(x)=(\boldsymbol{G}^1_n(x)\dots,\boldsymbol{G}^{D_n}_n(x))$, $\widetilde{\boldsymbol{H}}^{(\mathcal{B})}_{n'}$ can be viewed as the evaluation of the following composite polynomial at point $x=\alpha_{n'}$.

$$\psi(x)=\sum_{n=1}^{N}\sum_{i=1}^{D_n}\boldsymbol{F}^{i,(\mathcal{B})}_n(x)\boldsymbol{G}^i_n(x). \quad (6)$$

The server interpolates $\psi(x)$ from the received results $(\widetilde{\boldsymbol{H}}^{(\mathcal{B})}_{n'})_{n'\in\mathcal{U}}$, and evaluates it at $\beta_1,\dots,\beta_K$ to recover the summation of the embedding segments $\sum_{n=1}^{N}\overline{\boldsymbol{H}}^{(\mathcal{B})}_{n,1},\dots,\sum_{n=1}^{N}\overline{\boldsymbol{H}}^{(\mathcal{B})}_{n,K}$, where $\sum_{n=1}^{N}\overline{\boldsymbol{H}}^{(\mathcal{B})}_{n,k}=\psi(\beta_k)=\sum_{n=1}^{N}\sum_{i=1}^{D_n}\overline{\boldsymbol{X}}^{i,(\mathcal{B})}_{n,k}\overline{\boldsymbol{W}}^i_n$. The server horizontally stacks these summed segments to obtain the summation $\overline{\boldsymbol{H}}^{(\mathcal{B})}$ of local embeddings. Note that the overall batch size of $\overline{\boldsymbol{H}}^{(\mathcal{B})}$ is $K|\mathcal{B}|$.

Dequantization. The server maps $\overline{\boldsymbol{H}}^{(\mathcal{B})}$ back to the real domain to obtain an approximation of the average embedding $\boldsymbol{H}^{(\mathcal{B})}$ via applying the following dequantization function $\varphi:\mathbb{F}_p\to\mathbb{R}$ element-wise on $\overline{\boldsymbol{H}}^{(\mathcal{B})}$.

$$\varphi(x)=\begin{cases}\frac{1}{N}\cdot 2^{-(l_x+l_w)}\cdot x, & \text{if } 0\le x<\frac{p-1}{2}\\ \frac{1}{N}\cdot 2^{-(l_x+l_w)}\cdot(x-p), & \text{if } \frac{p-1}{2}\le x<p\end{cases}. \quad (7)$$

Next, server continues forward-backward propagation to update the central model $\boldsymbol{W}_0$. The server also computes $\nabla_{\boldsymbol{H}^{(\mathcal{B})}}L$, and broadcasts it to all clients. With $\nabla_{\boldsymbol{H}^{(\mathcal{B})}_n}L=\frac{1}{N}\nabla_{\boldsymbol{H}^{(\mathcal{B})}}L$, client n computes the gradient $\nabla_{\boldsymbol{W}_n}L=\nabla_{\boldsymbol{H}^{(\mathcal{B})}_n}L\cdot\nabla_{\boldsymbol{W}_n}\boldsymbol{H}^{(\mathcal{B})}_n$, and updates its local model $\boldsymbol{W}_n$.

5. Theoretical Analyses

5.1. Straggler resilience and privacy analysis

Theorem 5.1 (Straggler resilience). *The summation of local embeddings of all clients, i.e,* $\overline{\boldsymbol{H}}^{(\mathcal{B})}=\sum_{n=1}^{N}\overline{\boldsymbol{H}}^{(\mathcal{B})}_n$, *can be exactly recovered at the server, in the presence of up to* $N-2(K+T-1)-1$ *straggling clients.*

Proof. The server can exactly reconstruct $\psi(x)$, and hence the summation of local embeddings $\overline{\boldsymbol{H}}^{(\mathcal{B})}$, from the computation results of the non-straggling clients $(\widetilde{\boldsymbol{H}}^{(\mathcal{B})}_n)_{n\in\mathcal{U}}$, if $|\mathcal{U}|\ge\text{degree}(\psi(x))+1=2(K+T-1)+1$. Hence, the embedding aggregation process can tolerate up to $N-2(K+T-1)-1$ stragglers. □

Theorem 5.2 (Privacy against colluding clients). *Any subset of up to* T *colluding clients learn nothing about the local data and models of the other clients. More precisely, for any* $\mathcal{T}\subset[N]$ *with* $|\mathcal{T}|\le T$, *the mutual information* $I\left((\widetilde{\boldsymbol{X}}_n,\widetilde{\boldsymbol{W}}_n)_{n\in\mathcal{T}};(\overline{\boldsymbol{X}}_n,\overline{\boldsymbol{W}}_n)_{n\in[N]\setminus\mathcal{T}}\right)$ *equals zero.*

Proof. As the local data and models are secret shared using LCC, their privacy against T colluding clients follows the T-privacy guarantee of LCC construction (Theorem 1 in (Yu et al., 2019)). For completeness, we give a detailed proof in Appendix A. □

[2] WLOG, we assume that all clients successfully share their models with all other clients. In a more general scenario where each client may not be able to communicate with every other client, we can consider a subset $\mathcal{S}$ of clients who have successfully shared their models with a subset $\mathcal{R}$ of clients, and the proposed FedVS protocol can be used to compute the aggregated embedding $\sum_{n\in\mathcal{S}}\boldsymbol{H}^{(\mathcal{B})}_n$ from the uploaded results of clients in $\mathcal{R}$.

Theorem 5.3 (Privacy against curious server). *For each $n \in [N]$, the server learns nothing about the private data and the local model of client n, from its uploaded computation result. That is, the mutual information $I\left(\widetilde{\boldsymbol{H}}_n^{(\mathcal{B})}; (\overline{\boldsymbol{X}}_n, \overline{\boldsymbol{W}}_n)\right)$ equals zero.*

Proof. We know from the privacy guarantee of LCC that the secret shares of input data and model parameters at client n, i.e., $(\widetilde{\boldsymbol{X}}_n, \widetilde{\boldsymbol{W}}_n)$, reveal no information about its private data and model $(\overline{\boldsymbol{X}}_n, \overline{\boldsymbol{W}}_n)$. Moreover, as the output $\widetilde{\boldsymbol{H}}_n^{(\mathcal{B})}$ of client n is computed from $(\widetilde{\boldsymbol{X}}_n, \widetilde{\boldsymbol{W}}_n)$, i.e., $(\overline{\boldsymbol{X}}_n, \overline{\boldsymbol{W}}_n) \to (\widetilde{\boldsymbol{X}}_n, \widetilde{\boldsymbol{W}}_n) \to \widetilde{\boldsymbol{H}}_n^{(\mathcal{B})}$ forms a Markov chain, and we have $I\left(\widetilde{\boldsymbol{H}}_n^{(\mathcal{B})}; (\overline{\boldsymbol{X}}_n, \overline{\boldsymbol{W}}_n)\right) \le I\left((\widetilde{\boldsymbol{X}}_n, \widetilde{\boldsymbol{W}}_n); (\overline{\boldsymbol{X}}_n, \overline{\boldsymbol{W}}_n)\right) = 0$ by data processing inequality. □

Theorem 5.3 implies that in FedVS, local data and model of an individual client is *perfectly* secure against the server, which completely mitigates *any* privacy leakage from data inference and model stealing attacks on a client's output.

5.2. Convergence analysis

Since the rounding operation can be performed on both training and test data, FedVS can be considered to optimize the model parameters on the rounded data, which is denoted as $(\boldsymbol{x}'^{(m)}, \boldsymbol{y}^{(m)}), m \in [M]$. That is, we consider the following optimization problem for $\boldsymbol{W} = (\boldsymbol{W}_n)_{n=0}^{N}$.

$$\min_{\boldsymbol{W}} F(\boldsymbol{W}) \triangleq \frac{1}{M}\sum_{m=1}^{M} \ell(\boldsymbol{W}; (\boldsymbol{x}'^{(m)}, \boldsymbol{y}^{(m)})) = \frac{1}{M}\sum_{m=1}^{M} f_m(\boldsymbol{W}).$$

In round r of FedVS, for a sampled data batch $\mathcal{B}$, the server and the clients update their models as $\boldsymbol{W}_n^{r+1} = \boldsymbol{W}_n^r - \eta_n \nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r), \forall n \in \{0, 1, \ldots, N\}$, where $F_{\mathcal{B}}(\cdot) = \frac{1}{|\mathcal{B}|}\sum_{m \in \mathcal{B}} f_m(\cdot)$. Here we have $\widehat{\boldsymbol{W}}_0^r = \boldsymbol{W}_0^r$, as the central model is not rounded during forward propogation; for each client n, $\widehat{\boldsymbol{W}}_n^r = Q_{stoc}(\boldsymbol{W}_n^r) = 2^{-l_w} \cdot \mathrm{Round}_{stoc}(2^{l_w} \cdot \boldsymbol{W}_n^r)$.

We first make the following assumptions to facilitate our convergence analysis.

Assumption 1 (Variance-bounded stochastic rounding): There exists a constant $\gamma > 0$ such that $\forall z \in \mathbb{R}$, the operator $Q_{stoc}(.)$ satisfies $\mathbb{E}\left[\|Q_{stoc}(z) - z\|^2\right] \le \gamma^2 z^2$.

Assumption 2 (Lipschitz Smoothness): For any input $\boldsymbol{u}, \boldsymbol{v}$, there exists a constant $L > 0$, such that for all $m \in [M]$, the function f_m satisfies $\forall n \in \{0, 1, \ldots, N\}$, $\|\nabla_n f_m(\boldsymbol{u}) - \nabla_n f_m(\boldsymbol{v})\| \le L\|\boldsymbol{u} - \boldsymbol{v}\|$.

Assumption 3 (Global minimum existance): There exists a globally optimal collection of model parameters $\boldsymbol{W}^*$, such that $F(\boldsymbol{W}) \ge F(\boldsymbol{W}^*) > -\infty$, for all $\boldsymbol{W}$.

Assumption 4 (Bounded model parameters): The norm of the collection of all model parameters $\|\boldsymbol{W}\|$ is upper bounded by some constant σ.

We give the convergence result of FedVS in the following theorem, whose proof can be found in Appendix B.

Theorem 5.4 (Convergence of FedVS). *Under Assumption 1-4, when the learning rate $\eta_n = \frac{3}{4L}\frac{1}{\sqrt{R}}, \forall n \in \{0, 1, \ldots, N\}$, after R rounds of FedVS, with probability at least $1 - \delta$ we have:*

$$\frac{1}{R}\sum_{r=0}^{R-1}\mathbb{E}\Big(\sum_{n=0}^{N}\|\nabla_n F(\boldsymbol{W}^r)\|^2\Big) \le \frac{16L}{9\sqrt{R}}(F(\boldsymbol{W}^0) - F(\boldsymbol{W}^*)) + \frac{\sum_{n=0}^{N}(2L^2\gamma^2\sigma^2 + 2V_n)}{\sqrt{R}} = \mathcal{O}\left(\frac{1}{\sqrt{R}}\right),$$

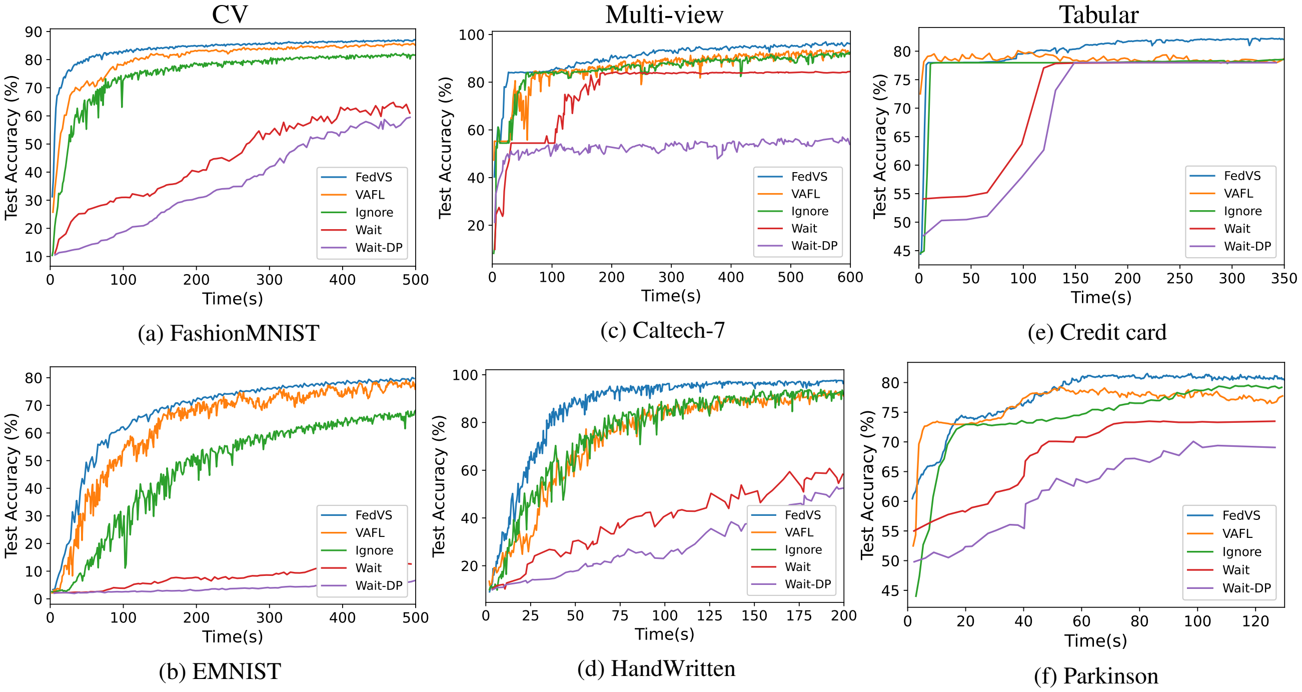
where $V_n = \frac{32L^2(\log(2p_n/\delta) + \frac{1}{4})}{|\mathcal{B}|}$, and p_n is dimension of $\boldsymbol{W}_n$.

5.3. Complexity analysis

Computation and communication costs for data sharing. Before training starts, each client secret shares its local data using LCC. Given that evaluating a polynomial of degree $K + T - 1$ at N points can be done using $\mathcal{O}(N \log^2 N)$ operations in $\mathbb{F}_p$ (Von Zur Gathen & Gerhard, 2013), the computation load at client n to generate N shares is $\mathcal{O}(\frac{M d_n D_n}{K} N \log^2 N)$. The communication cost for client n to secret share its data is $\mathcal{O}(\frac{M d_n D_n N}{K})$. We note that these computation and communication overheads occur once before the training starts, and become less relevant as the number of training rounds increases.

Computation and communication costs for a training round. In each training round, each client n first needs to secret share its local model, which takes a computation load of $\mathcal{O}(d_n h D_n N \log^2 N)$ and a communication load of $\mathcal{O}(d_n h D_n N)$. Next, for the sampled data batch $\mathcal{B}$ of size $|\mathcal{B}| \le \frac{M}{K}$, client n performs embedding computation as in (5) with a computation load of $\mathcal{O}(|\mathcal{B}| h \sum_{i=1}^{N} d_i D_i)$, and sends the computed results to the server with a communication load of $\mathcal{O}(|\mathcal{B}| h)$. Note that while according to Theorem 5.1 a smaller partition paramter K allows to tolerate more stragglers, the load of embedding computation is also higher. We stress that in FedVS, the loads of computing and communicating (coded) embeddings are identical across all clients, further alleviating the straggler effect caused by imbalanced data and model dimensions.

Server decodes the embedding aggregation from results of $R = 2(K + T - 1) + 1$ non-straggling clients, via interpolating $\psi(x)$ in (6) with a computation cost of $\mathcal{O}(|\mathcal{B}| h R \log^2 R)$.



(a) FashionMNIST

(c) Caltech-7

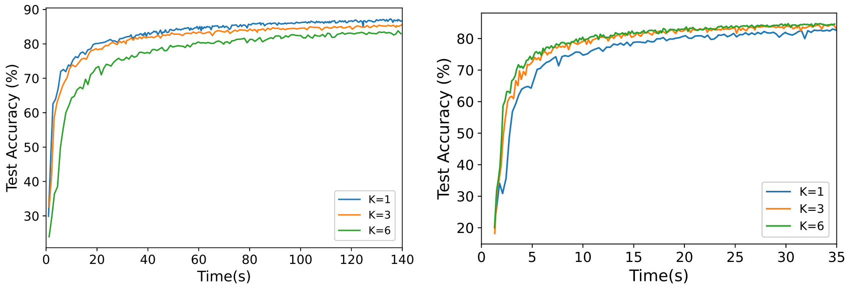
(e) Credit card

(b) EMNIST

(d) HandWritten

(f) Parkinson

Figure 2. Test accuracies using different straggler handling and privacy protection methods on different datasets.



(a) Delay dominant.

(b) Computation dominant.

Figure 3. Test accuracies of FedVS on FashionMNIST using different K.

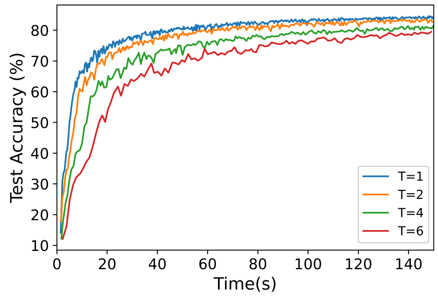


Figure 4. Test accuracies of FedVS on FashionMNIST using different T.

6. Experimental Evaluations

We carry out split VFL experiments on three types of six real-world datasets, and compare the performance of FedVS in straggler mitigation and privacy protection with four baselines. All experiments are performed on a single machine using four NVIDIA GeForce RTX 3090 GPUs.

6.1. Datasets

We consider three types of data, and select two datasets from each type. For tabular datasets Parkinson (Sakar et al., 2019) and Credit card (Yeh & Lien, 2009), and computer vision (CV) datasets EMNIST (Cohen et al., 2017) and FashionMNIST (Xiao et al., 2017b), we evenly partition the features of each data sample across the clients. For the multi-view datasets Handwritten (Dua & Graff, 2017) and Caltech-7 (Li et al., 2022), each client holds one view of each data sample. We provide descriptions of the datasets, number of clients considered for each dataset, employed model architectures, and training parameters in Appendix C.

6.2. Experiment settings

Baselines. We consider the following four baseline methods for straggler handling and privacy protection. 1) *Wait*: Server waits for all clients (including stragglers) for embedding aggregation; 2) *Ignore*: Server ignores the stragglers, and proceeds with aggregating embeddings from non-stragglers; 3) *VAFL* (Chen et al., 2020): Server asynchronously receives embeddings and updates model parameters; 4) *Wait-DP*: To utilize differential privacy to protect clients' data and model privacy, as in (Thapa et al., 2022), a calibrated noise is added to the output (e.g., embedding) of a network layer at each client, and the server waits to aggregate all clients' perturbed embeddings.

Delay pattern. We add artificial delays to the clients' computations to simulate the effect of stragglers. Before the clients upload their computed embeddings, 50% of them add a random delay sampled from an exponential distribution with a mean of 0.2s. The other 50% are modelled as stragglers, whose delays are sampled from exponential distributions with incremental means, i.e., $1 + \frac{2i}{N}, i \in [\frac{N}{2}]$. Besides, the straggler effect in the model sharing phase of FedVS is also simulated by adding an exponential delay at each client, whose mean, according to the analysis of computation costs, is $1/|\mathcal{B}|$ of the corresponding delay's mean for embedding uploading.

Parameter settings. For Wait-DP, we set the privacy budget ϵ' to 10. For FedVS, we optimize the rate of convergence over the partition parameter K for each dataset. The privacy parameter of FedVS is set to $T = 1$. To simulate communication delays, a network bandwidth of 300Mbps, as measured in (So et al., 2022) for AWS EC2 cloud computing environment, is assumed for the server and all clients. Each experiment is repeated 5 times and the average accuracies are reported.

6.3. Results

Comparisons with baselines. As shown in Figure 2, for CV and multi-view datasets, FedVS outperforms all baselines in test accuracy at all times. For tabular datasets, VAFL and Ignore converge quickly at the beginning and are eventually outperformed by FedVS. For privacy protection, inserting DP noises in Wait-DP hurts the accuracies for all datasets. In sharp contrast, FedVS protects data and model privacy without performance loss.

Optimization of partition parameter. We further explore the optimal choices of the partition parameter K for FedVS under different straggler patterns. Specifically, we consider two delay patterns depending on whether the mean of clients' added delays is greater than the local computation time at a single client. As shown in Figure 3(a), when the mean delay is greater than the computation time, stragglers cause major performance bottleneck, and it is preferable to use a smaller K to tolerate more stragglers. On the other hand, when the local computation time dominates the delay caused by stragglers, Figure 3(b) indicates that it is optimal to choose a larger K to minimize local computation load.

Privacy-performance tradeoff. For a larger privacy parameter T, the privacy guarantee of FedVS becomes stronger as it protects data and model privacy from T colluding clients. However, as shown in Figure 4, its performance suffers as it tolerates less number of stragglers.

7. Conclusion

We propose FedVS, a synchronous split VFL framework that simultaneously addresses the problems of straggling clients and privacy leakage. Through efficient secret sharing of data and model parameters and descryption on the computation shares, FedVS losslessly aggregates embeddings from all clients, in presence of a certain number of stragglers; and simultaneously provides information-theoretic privacy against the curious server and a certain number of colluding clients. Extensive experiments on various VFL tasks and datasets further demonstrate the superiority of FedVS in straggler mitigation and privacy protection over baseline methods.

Acknowledgement

This work is in part supported by the National Nature Science Foundation of China (NSFC) Grant 62106057, Foshan HKUST Projects FSUST20-FYTRI04B, and Guangzhou Science and Technology City-University Joint Project 2023A03J0151.

References

Bell, J. H., Bonawitz, K. A., Gascón, A., Lepoint, T., and Raykova, M. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1253–1269, 2020.

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1175–1191, 2017.

Cai, S., Chai, D., Yang, L., Zhang, J., Jin, Y., Wang, L., Guo, K., and Chen, K. Secure forward aggregation for vertical federated neural networks. *arXiv preprint arXiv:2207.00165*, 2022.

Castiglia, T., Wang, S., and Patterson, S. Flexible vertical federated learning with heterogeneous parties. *arXiv preprint arXiv:2208.12672*, 2022.

Ceballos, I., Sharma, V., Mugica, E., Singh, A., Roman, A., Vepakomma, P., and Raskar, R. Splitnn-driven vertical partitioning. *CoRR*, abs/2008.04137, 2020. URL `https://arxiv.org/abs/2008.04137`.

Chai, D., Wang, L., Chen, K., and Yang, Q. Secure federated matrix factorization. *IEEE Intelligent Systems*, 36(5):11–20, 2020.

Chai, Z., Chen, Y., Anwar, A., Zhao, L., Cheng, Y., and Rangwala, H. Fedat: a high-performance and communication-efficient federated learning system with asynchronous tiers. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16, 2021.

Chen, T., Jin, X., Sun, Y., and Yin, W. VAFL: a method of vertical asynchronous federated learning. *CoRR*, abs/2007.06081, 2020. URL https://arxiv.org/abs/2007.06081.

Cheng, K., Fan, T., Jin, Y., Liu, Y., Chen, T., Papadopoulos, D., and Yang, Q. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6): 87–98, 2021.

Cheng, Y., Liu, Y., Chen, T., and Yang, Q. Federated learning for privacy-preserving AI. *Communications of the ACM*, 63(12):33–36, 2020.

Choi, B., Sohn, J.-y., Han, D.-J., and Moon, J. Communication-computation efficient secure aggregation for federated learning. *arXiv preprint arXiv:2012.05433*, 2020.

Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pp. 2921–2926. IEEE, 2017.

Dhakal, S., Prakash, S., Yona, Y., Talwar, S., and Himayat, N. Coded federated learning. In *2019 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6. IEEE, 2019.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Erdogan, E., Kupcu, A., and Cicek, A. E. Unsplit: Data-oblivious model inversion, model stealing, and label inference attacks against split learning. *arXiv preprint arXiv:2108.09033*, 2021.

Feng, S. and Yu, H. Multi-participant multi-class vertical federated learning. *CoRR*, abs/2001.11154, 2020. URL https://arxiv.org/abs/2001.11154.

Fu, C., Zhang, X., Ji, S., Chen, J., Wu, J., Guo, S., Zhou, J., Liu, A. X., and Wang, T. Label inference attacks against vertical federated learning. In *31st USENIX Security Symposium (USENIX Security 22), Boston, MA*, 2022.

Gu, B., Xu, A., Huo, Z., Deng, C., and Huang, H. Privacy-preserving asynchronous vertical federated learning algorithms for multiparty collaborative learning. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2021. doi: 10.1109/TNNLS.2021.3072238.

Hardy, S., Henecka, W., Ivey-Law, H., Nock, R., Patrini, G., Smith, G., and Thorne, B. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

Hu, Y., Niu, D., Yang, J., and Zhou, S. Fdml: A collaborative machine learning framework for distributed features. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2232–2240, 2019.

Huba, D., Nguyen, J., Malik, K., Zhu, R., Rabbat, M., Yousefpour, A., Wu, C.-J., Zhan, H., Ustinov, P., Srinivas, H., et al. Papaya: Practical, private, and scalable federated learning. *Proceedings of Machine Learning and Systems*, 4:814–832, 2022.

Jahani-Nezhad, T., Maddah-Ali, M. A., Li, S., and Caire, G. Swiftagg: Communication-efficient and dropout-resistant secure aggregation for federated learning with worst-case security guarantees. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pp. 103–108, 2022a. doi: 10.1109/ISIT50566.2022.9834750.

Jahani-Nezhad, T., Maddah-Ali, M. A., Li, S., and Caire, G. Swiftagg+: Achieving asymptotically optimal communication load in secure aggregation for federated learning. *arXiv preprint arXiv:2203.13060*, 2022b.

Jin, X., Chen, P.-Y., Hsu, C.-Y., Yu, C.-M., and Chen, T. Cafe: Catastrophic data leakage in vertical federated learning. *Advances in Neural Information Processing Systems*, 34:994–1006, 2021.

Kairouz, P., Liu, Z., and Steinke, T. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *International Conference on Machine Learning*. PMLR, 2021.

Lee, J., Sun, J., Wang, F., Wang, S., Jun, C.-H., Jiang, X., et al. Privacy-preserving patient similarity learning in a federated environment: development and analysis. *JMIR medical informatics*, 6(2):e7744, 2018.

Li, F.-F., Andreeto, M., Ranzato, M., and Perona, P. Caltech 101, April 2022.

Li, M., Chen, Y., Wang, Y., and Pan, Y. Efficient asynchronous vertical federated learning via gradient prediction and double-end sparse compression. In *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 291–296. IEEE, 2020.

Li, Q., Wen, Z., Wu, Z., Hu, S., Wang, N., Li, Y., Liu, X., and He, B. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 2021a.

Li, X., Qu, Z., Tang, B., and Lu, Z. Stragglers are not disaster: A hybrid federated learning algorithm with delayed gradients. *arXiv preprint arXiv:2102.06329*, 2021b.

Liu, L., Gu, R., and Hu, X. Ladder polynomial neural networks. *CoRR*, abs/2106.13834, 2021. URL https://arxiv.org/abs/2106.13834.

Liu, Z., Guo, J., Lam, K.-Y., and Zhao, J. Efficient dropout-resilient aggregation for privacy-preserving machine learning. *IEEE Transactions on Information Forensics and Security*, 2022a.

Liu, Z., Guo, J., Yang, W., Fan, J., Lam, K.-Y., and Zhao, J. Privacy-preserving aggregation in federated learning: A survey. *arXiv preprint arXiv:2203.17005*, 2022b.

Livni, R., Shalev-Shwartz, S., and Shamir, O. On the computational efficiency of training neural networks. *Advances in neural information processing systems*, 27, 2014.

Luo, X., Wu, Y., Xiao, X., and Ooi, B. C. Feature inference attack on model predictions in vertical federated learning. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 181–192. IEEE, 2021.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Nguyen, J., Malik, K., Zhan, H., Yousefpour, A., Rabbat, M., Malek, M., and Huba, D. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*, pp. 3581–3607. PMLR, 2022.

Park, J., Han, D.-J., Choi, M., and Moon, J. Sageflow: Robust federated learning against both stragglers and adversaries. *Advances in Neural Information Processing Systems*, 34:840–851, 2021.

Prakash, S., Dhakal, S., Akdeniz, M. R., Yona, Y., Talwar, S., Avestimehr, S., and Himayat, N. Coded computing for low-latency federated learning over wireless edge networks. *IEEE Journal on Selected Areas in Communications*, 39(1):233–250, 2020.

Ramezani, M., Cong, W., Mahdavi, M., Sivasubramaniam, A., and Kandemir, M. Gcn meets gpu: Decoupling “when to sample”from “how to sample”. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 18482–18492. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/d714d2c5a796d5814c565d78dd16188d-Paper.pdf.

Reisizadeh, A., Tziotis, I., Hassani, H., Mokhtari, A., and Pedarsani, R. Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity. *IEEE Journal on Selected Areas in Information Theory*, 2022.

Sakar, C. O., Serbes, G., Gunduz, A., Tunc, H. C., Nizam, H., Sakar, B. E., Tutuncu, M., Aydin, T., Isenkul, M. E., and Apaydin, H. A comparative analysis of speech signal processing algorithms for parkinson’s disease classification and the use of the tunable q-factor wavelet transform. *Applied Soft Computing*, 74:255–263, 2019.

Schlegel, R., Kumar, S., Rosnes, E., et al. Codedpaddedfl and codedsecagg: Straggler mitigation and secure aggregation in federated learning. *arXiv preprint arXiv:2112.08909*, 2021.

Shao, J., Sun, Y., Li, S., and Zhang, J. Dres-fl: Dropout-resilient secure federated learning for non-iid clients via secret data sharing. *Advances in Neural Information Processing Systems*, 2022.

Shi, H., Xu, Y., Jiang, Y., Yu, H., and Cui, L. Efficient asynchronous multi-participant vertical federated learning. *IEEE Transactions on Big Data*, 2022.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

So, J., Güler, B., and Avestimehr, A. S. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory*, 2(1):479–489, 2021.

So, J., He, C., Yang, C.-S., Li, S., Yu, Q., E Ali, R., Guler, B., and Avestimehr, S. Lightsecagg: a lightweight and versatile design for secure aggregation in federated learning. *Proceedings of Machine Learning and Systems*, 4:694–720, 2022.

Sun, Y., Shao, J., Li, S., Mao, Y., and Zhang, J. Stochastic coded federated learning with convergence and privacy guarantees. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pp. 2028–2033, 2022a.

Sun, Y., Shao, J., Mao, Y., Li, S., and Zhang, J. Stochastic coded federated learning: Theoretical analysis and incentive mechanism design. *arXiv preprint arXiv:2211.04132*, 2022b.

Thapa, C., Arachchige, P. C. M., Camtepe, S., and Sun, L. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8485–8493, 2022.

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*, 2019.

Truex, S., Liu, L., Chow, K.-H., Gursoy, M. E., and Wei, W. Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, pp. 61–66, 2020.

van Dijk, M., Nguyen, N. V., Nguyen, T. N., Nguyen, L. M., Tran-Dinh, Q., and Nguyen, P. H. Asynchronous federated learning with reduced number of rounds and with differential privacy from less aggregated gaussian noise. *arXiv preprint arXiv:2007.09208*, 2020.

Vepakomma, P., Gupta, O., Swedish, T., and Raskar, R. Split learning for health: Distributed deep learning without sharing raw patient data. *CoRR*, abs/1812.00564, 2018. URL http://arxiv.org/abs/1812.00564.

Von Zur Gathen, J. and Gerhard, J. *Modern computer algebra*. Cambridge university press, 2013.

Wang, C., Liang, J., Huang, M., Bai, B., Bai, K., and Li, H. Hybrid differentially private federated learning on vertically partitioned data. *arXiv preprint arXiv:2009.02763*, 2020.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q., and Poor, H. V. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

Wei, K., Li, J., Ma, C., Ding, M., Wei, S., Wu, F., Chen, G., and Ranbaduge, T. Vertical federated learning: Challenges, methodologies and experiments. *arXiv preprint arXiv:2202.04309*, 2022.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017a.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017b.

Xie, C., Koyejo, S., and Gupta, I. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*, 2019.

Xu, D., Yuan, S., and Wu, X. Achieving differential privacy in vertically partitioned multiparty learning. In *2021 IEEE International Conference on Big Data (Big Data)*, pp. 5474–5483. IEEE, 2021.

Yang, S., Ren, B., Zhou, X., and Liu, L. Parallel distributed logistic regression for vertical federated learning without third-party coordinator. *CoRR*, abs/1911.09824, 2019a. URL http://arxiv.org/abs/1911.09824.

Yang, S., Ren, B., Zhou, X., and Liu, L. Parallel distributed logistic regression for vertical federated learning without third-party coordinator. *arXiv preprint arXiv:1911.09824*, 2019b.

Yeh, I.-C. and Lien, C.-h. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert systems with applications*, 36(2):2473–2480, 2009.

Yu, Q., Li, S., Raviv, N., Kalan, S. M. M., Soltanolkotabi, M., and Avestimehr, S. A. Lagrange coded computing: Optimal design for resiliency, security, and privacy. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1215–1225. PMLR, 2019.

Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., and Liu, Y. BatchCrypt: Efficient homomorphic encryption for Cross-Silo federated learning. In *2020 USENIX annual technical conference (USENIX ATC 20)*, pp. 493–506, 2020.

Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., and Gao, Y. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021a. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2021.106775. URL https://www.sciencedirect.com/science/article/pii/S0950705121000381.

Zhang, Q., Gu, B., Deng, C., and Huang, H. Secure bilevel asynchronous vertical federated learning with backward updating. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10896–10904, 2021b.

Appendix

A. Proof of Theorem 5.2

Here we prove information-theoretic data privacy against T colluding clients, and the proof for model privacy follows the similar steps.

WLOG, let us consider the first T clients colluding to infer private data $\overline{\boldsymbol{X}}_n$ of client $n > T$. We have from (2) that the secret shares of $\overline{\boldsymbol{X}}_n$ at the first T clients are

$$\underbrace{\begin{pmatrix} \widetilde{\boldsymbol{X}}_{n,1} \\ \widetilde{\boldsymbol{X}}_{n,2} \\ \vdots \\ \widetilde{\boldsymbol{X}}_{n,T} \end{pmatrix}}_{\widetilde{\boldsymbol{X}}_{n,T}} = \underbrace{\begin{pmatrix} a_{1,1} & \dots & a_{1,K} \\ a_{2,1} & \dots & a_{2,K} \\ \vdots & \dots & \vdots \\ a_{T,1} & \dots & a_{T,K} \end{pmatrix}}_{\boldsymbol{A}_1} \underbrace{\begin{pmatrix} \overline{\boldsymbol{X}}_{n,1} \\ \overline{\boldsymbol{X}}_{n,2} \\ \vdots \\ \overline{\boldsymbol{X}}_{n,K} \end{pmatrix}}_{\overline{\boldsymbol{X}}_n} + \underbrace{\begin{pmatrix} a_{1,K+1} & \dots & a_{1,K+T} \\ a_{2,K+1} & \dots & a_{2,K+T} \\ \vdots & \dots & \vdots \\ a_{T,K+1} & \dots & a_{T,K+T} \end{pmatrix}}_{\boldsymbol{A}_2} \underbrace{\begin{pmatrix} \boldsymbol{Z}_{n,K+1} \\ \boldsymbol{Z}_{n,K+2} \\ \vdots \\ \boldsymbol{Z}_{n,K+T} \end{pmatrix}}_{\boldsymbol{Z}_n} \tag{8}$$

Here $a_{n',k} = \prod_{\ell\in[K+T]\setminus\{k\}} \frac{\alpha_{n'}-\beta_\ell}{\beta_k-\beta_\ell}$, for all $k \in [K+T]$.

As $\boldsymbol{Z}_n$ is uniformly random in $\mathbb{F}_p^{\frac{TM}{K}\times d_nD_n}$ and the matrix $\boldsymbol{A}_2$ comprised of Lagrange coefficients has full rank, $\boldsymbol{A}_2\boldsymbol{Z}_n$ is also uniformly random in $\mathbb{F}_p^{\frac{TM}{K}\times d_nD_n}$. Now, for any $\boldsymbol{M} \in \mathbb{F}_p^{\frac{TM}{K}\times d_nD_n}$ and $\boldsymbol{N} \in \mathbb{F}_p^{M\times d_nD_n}$, we have

$$\text{Pr}[\widetilde{\boldsymbol{X}}_{n,T} = \boldsymbol{M}|\overline{\boldsymbol{X}}_n = \boldsymbol{N}] = \text{Pr}[\boldsymbol{A}_2\boldsymbol{Z}_n = \boldsymbol{M} - \boldsymbol{A}_1\boldsymbol{N}|\overline{\boldsymbol{X}}_n = \boldsymbol{N}] \tag{9}$$

$$\overset{(a)}{=} \text{Pr}[\boldsymbol{A}_2\boldsymbol{Z}_n = \boldsymbol{M} - \boldsymbol{A}_1\boldsymbol{N}] \tag{10}$$

$$\overset{(b)}{=} \frac{1}{p^{\frac{TMd_nD_n}{K}}}. \tag{11}$$

Here (a) is because that $\overline{\boldsymbol{X}}_n$ and $\boldsymbol{Z}_n$ are independent, and (b) is because that $\boldsymbol{A}_2\boldsymbol{Z}_n$ is uniformly random in $\mathbb{F}_p^{\frac{TM}{K}\times d_nD_n}$. Next, we have

$$\text{Pr}[\widetilde{\boldsymbol{X}}_{n,T} = \boldsymbol{M}] = \sum_{\boldsymbol{N}} \text{Pr}[\widetilde{\boldsymbol{X}}_{n,T} = \boldsymbol{M}|\overline{\boldsymbol{X}}_n = \boldsymbol{N}]\text{Pr}[\overline{\boldsymbol{X}}_n = \boldsymbol{N}] \tag{12}$$

$$= \sum_{\boldsymbol{N}} \frac{1}{p^{\frac{TMd_nD_n}{K}}} \text{Pr}[\overline{\boldsymbol{X}}_n = \boldsymbol{N}] = \frac{1}{p^{\frac{TMd_nD_n}{K}}}. \tag{13}$$

We have from the above that for any $\boldsymbol{M}$ and $\boldsymbol{N}$, $\text{Pr}[\widetilde{\boldsymbol{X}}_{n,T} = \boldsymbol{M}|\overline{\boldsymbol{X}}_n = \boldsymbol{N}] = \text{Pr}[\widetilde{\boldsymbol{X}}_{n,T} = \boldsymbol{M}] = \frac{1}{p^{\frac{TMd_nD_n}{K}}}$, and hence $\widetilde{\boldsymbol{X}}_{n,T}$ and $\overline{\boldsymbol{X}}_n$ are statistically independent. That is, the mutual information $I\left(\widetilde{\boldsymbol{X}}_{n,T}; \overline{\boldsymbol{X}}_n\right) = 0$. As this holds for all $n > T$, we have $I(\widetilde{\boldsymbol{X}}_{T+1,T}, \widetilde{\boldsymbol{X}}_{T+2,T}, \dots, \widetilde{\boldsymbol{X}}_{N,T}; \overline{\boldsymbol{X}}_{T+1}, \overline{\boldsymbol{X}}_{T+2}, \dots, \overline{\boldsymbol{X}}_N) = 0$.

B. Proof of Theorem 5.4

From Lemma 10 in (Ramezani et al., 2020), we state the following lemma that bounds the difference between the gradients of the losses computed from a sampled batch and all training data.

Lemma B.1. *Consider mini-batch function $\nabla_n F_{\mathcal{B}}(\boldsymbol{W}) \in \mathbb{R}^{p_n}, n \in \{0, 1, \dots, N\}$, which satisfies $\mathbb{E}[\nabla_n F_{\mathcal{B}}(\boldsymbol{W})] = \nabla_n F(\boldsymbol{W})$. For $\epsilon < 2L$, we have with probability at least $1 - \delta$ that:*

$$\|\nabla_n F_{\mathcal{B}}(\boldsymbol{W}) - \nabla_n F(\boldsymbol{W})\|^2 \leq \frac{32L^2(\log(2p_n/\delta) + \frac{1}{4})}{|\mathcal{B}|}. \tag{14}$$

Proof. The proof refers to (Ramezani et al., 2020). □

Then we provide lemma B.2 to bound the loss function in each round r as follows:

Lemma B.2. *Under Assumption 2, for each round r, it follows that*

$$F(\boldsymbol{W}^{r+1}) \leq F(\boldsymbol{W}^r) + \sum_{n=0}^{N}(L\eta_n^2 - \frac{3}{2}\eta_n)\|\nabla_n F(\boldsymbol{W}^r)\|^2 + \sum_{n=0}^{N} L\eta_n^2 \|\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r) - \nabla_n F(\boldsymbol{W}^r)\|^2, \tag{15}$$

Proof. From Assumption 1, we can derive that:

$$\begin{aligned} F(\boldsymbol{W}^{r+1}) =& F\left(\boldsymbol{W}_0^r - \eta_0 \nabla_0 F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r), \dots, \boldsymbol{W}_N^r - \eta_N \nabla_N F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r)\right) \\ \leq& F(\boldsymbol{W}^r) - \sum_{n=0}^{N} \langle \nabla_n F(\boldsymbol{W}^r), \eta_n(\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r) - \nabla_n F(\boldsymbol{W}^r) + \nabla_n F(\boldsymbol{W}^r))\rangle + \sum_{n=0}^{N} \frac{L}{2}\eta_n^2 \|\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r)\|^2 \\ =& F(\boldsymbol{W}^r) - \sum_{n=0}^{N} \eta_n \|\nabla_n F(\boldsymbol{W}^r)\|^2 - \sum_{n=0}^{N} \eta_n \langle \nabla_n F(\boldsymbol{W}^r), (\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r) - \nabla_n F(\boldsymbol{W}^r))\rangle \\ &+ \sum_{n=0}^{N} \frac{L}{2}\eta_n^2 \|\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r)\|^2. \end{aligned} \tag{16}$$

Note that we have:

$$\begin{aligned} \|\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r)\|^2 =& \|\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r) - \nabla_n F(\boldsymbol{W}^r) + \nabla_n F(\boldsymbol{W}^r)\|^2 \\ =& \|\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r) - \nabla_n F(\boldsymbol{W}^r)\|^2 + \|\nabla_n F(\boldsymbol{W}^r)\|^2 \\ &+ 2\langle \nabla_n F(\boldsymbol{W}^r), \nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r) - \nabla_n F(\boldsymbol{W}^r)\rangle, \forall n \in [N]. \end{aligned} \tag{17}$$

Combining (16) and (17), we have:

$$\begin{aligned} F(\boldsymbol{W}^{r+1}) &\leq F(\boldsymbol{W}^r) + \sum_{n=0}^{N}(L\eta_n^2 - \frac{3}{2}\eta_n)\|\nabla_n F(\boldsymbol{W}^r)\|^2 + \sum_{n=0}^{N}(L\eta_n^2 - \frac{1}{2}\eta_n)\|\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r) - \nabla_n F(\boldsymbol{W}^r)\|^2 \\ &\leq F(\boldsymbol{W}^r) + \sum_{n=0}^{N}(L\eta_n^2 - \frac{3}{2}\eta_n)\|\nabla_n F(\boldsymbol{W}^r)\|^2 + \sum_{n=0}^{N} L\eta_n^2 \|\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r) - \nabla_n F(\boldsymbol{W}^r)\|^2, \end{aligned} \tag{18}$$

which completes the proof of lemma B.2. □

Proof of Theorem 5.4:

Considering the stochastic rounding on clients' model parameters, from Assumption 1, 2, 4 and Lemma B.1, we can derive the following inequality with probability at least $1-\delta, \forall n \in \{0, 1, \dots, N\}$:

$$\begin{aligned} \|\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r) - \nabla_n F(\boldsymbol{W}^k)\|^2 &\leq 2\|\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r) - \nabla_n F_{\mathcal{B}}(\boldsymbol{W}^r)\|^2 + 2\|\nabla_n F_{\mathcal{B}}(\boldsymbol{W}^r) - \nabla_n F(\boldsymbol{W}^r)\|^2 \\ &\leq 2L^2\|(\boldsymbol{W}_0^r, Q_{stoc}(\boldsymbol{W}_1^r, \dots, \boldsymbol{W}_N^r)) - \boldsymbol{W}^r\|^2 + 2\|\nabla_n F_{\mathcal{B}}(\boldsymbol{W}^r) - \nabla_n F(\boldsymbol{W}^r)\|^2 \\ &\leq 2L^2\|Q_{stoc}(\boldsymbol{W}_0^r, \boldsymbol{W}_1^r, \dots, \boldsymbol{W}_N^r) - \boldsymbol{W}^r\|^2 + 2\|\nabla_n F_{\mathcal{B}}(\boldsymbol{W}^r) - \nabla_n F(\boldsymbol{W}^r)\|^2 \\ &= 2L^2\gamma^2\sigma^2 + \frac{64L^2(\log(2p_n/\delta) + \frac{1}{4})}{|\mathcal{B}|} \\ &= 2L^2\gamma^2\sigma^2 + 2V_n, \end{aligned} \tag{19}$$

where $V_n = \frac{32L^2(\log(2p_n/\delta)+\frac{1}{4})}{|\mathcal{B}|}$.

When $\eta_n \leq \frac{3}{4L}, \forall n \in \{0, 1, \dots, N\}$ in Lemma B.2, the following inequality holds:

$$\sum_{n=0}^{N} \frac{3}{4}\eta_n \|\nabla_n F(\boldsymbol{W}^r)\|^2 \leq F(\boldsymbol{W}^r) - F(\boldsymbol{W}^{r+1}) + \sum_{n=0}^{N} L\eta_n^2 \|\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r) - \nabla_n F(\boldsymbol{W}^r)\|^2, \tag{20}$$

Under Assumption 3, taking expectation on both sides of (20) and adopting $\eta_n = \frac{3}{4L}\sqrt{\frac{1}{R}} \le \frac{3}{4L}, \forall n \in \{0, 1, \ldots, N\}$, the following holds with probability at least $1-\delta$:

$$\begin{aligned}
\frac{1}{R}\sum_{r=0}^{R-1}\mathbb{E}\left(\sum_{n=0}^{N}\|\nabla_n F(\boldsymbol{W}^r)\|^2\right) &\le \frac{F(\boldsymbol{W}^0)-F(\boldsymbol{W}^*)}{\frac{3}{4}\eta_n R} + \frac{\sum_{r=0}^{R-1}\mathbb{E}(\sum_{n=0}^{N} L\eta_n^2\|\nabla_n F_{\mathcal{B}}(\widehat{\boldsymbol{W}}^r)-\nabla_n F(\boldsymbol{W}^r)\|^2)]}{\frac{3}{4}\eta_n R} \\
&\le \frac{16L}{9\sqrt{R}}(F(\boldsymbol{W}^0)-F(\boldsymbol{W}^*)) + \frac{\sum_{n=0}^{N}(2L^2\gamma^2\sigma^2+2V_n)}{\sqrt{R}} \qquad (21) \\
&= \mathcal{O}\left(\frac{1}{\sqrt{R}}\right).
\end{aligned}$$

This completes the proof of Theorem 5.4.

C. Six datasets' descriptions, models and training details

Table 2. Dataset Descriptions.

	Tabular		Multi-view		CV	
	Parkinson	Credit card	Handwritten	Caltech-7	EMNIST	FashionMNIST
Number of samples	756	30,000	2000	1474	131,600	70,000
Feature size	754	24	649	3766	784	784
Number of classes	2	2	10	7	47	10

Parkinson: The dataset's features are biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD). Each feature is a particular voice measurement. The label is divided to 0 and 1, which represents PD and healthy people. There are 10 clients with vertically partitioned data. 70% of the data is regarded as training data and the remaining part is test data. Each client holds a 2-layer PN, and the server holds a network with 2 Linear-ReLU layers and 1 Linear-Sigmoid layer. The learning rate is set to 0.005. The batch size is 16.

Credit Card: The dataset is composed of information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005, which contains 24 attributes. The labels of the samples are biased since there are 78% of samples labeled as 0 and 22% of samples labeled as 1 to denote default payment. 11 clients equally hold vertically partitioned data. Each client holds a 2-layer PN, and the server holds a network with a Linear-BatchNorm-Linear-ReLU-BatchNorm-WeightNorm-Linear-Sigmoid structure. The learning rate is set to 0.01 and the batch size is set to 32.

FashionMNIST: It is an image dataset related to household goods. Each image sample is evenly partitioned across 28 clients. Each client holds a 1-layer PN, and the server holds a network with 2 Linear-ReLU layers and one Linear-Logsoftmax layer. The learning rate is set to 0.5 and batch size 256 is selected.

EMNIST: The EMNIST dataset is a set of handwritten character digits converted to a 28x28 pixel image format. There are 28 clients and the data is partitioned the same as FashionMNIST. The clients' model is a 2-layer PN, and the server model is the same as the above FashionMNIST server's model. The learning rate is 0.05 and the batch size is 512.

HandWritten: The dataset consists of features of handwritten numerals 0-9 extracted from a collection of Dutch utility maps. 200 patterns per class (for a total of 2,000 patterns) have been digitized in binary images. It consists of 6 views, pixel (PIX) of dimension 240, Fourier coefficients of dimension 76, profile correlations (FAC) of dimension 216, Zernike moments (ZER) of dimension 47, Karhunen-Loeve coefficients (KAR) of dimension 64 and morphological features (MOR) of dimension 6. Each client holds one view. The dataset is split to 60% as the train set and 40% as the test set. Each client holds a 2-layer PN, and the server holds a model with 2 Linear-ReLU layers and 1 Linear-Logsoftmax layer. The learning rate is 0.02 and the batch size is 8.

Caltech-7: Caltech-101 is an object recognition dataset containing 8677 images of 101 categories. 7 classes of Caltech 101 are selected, i.e., Face, Motorbikes, Dolla-Bill, Garfield, Snoopy, Stop-Signand Windsor-Chair. The dataset is composed of

6 views, each of which is held by a client. 80% of the dataset is used for training and 20% for testing. Each client holds a 2-layer PN. The server holds the same model structure as HandWritten. The learning rate is 0.02 and the batch size is 8.