Efficient and Privacy-Preserving Feature Importance-based Vertical Federated Learning

Anran Li, Jiahui Huang, Ju Jia, Hongyi Peng, Lan Zhang, Luu Anh Tuan, Han Yu, and Xiang-Yang Li *Fellow, ACM & IEEE*

Abstract—Vertical Federated Learning (VFL) enables multiple data owners, each holding a different subset of features about a largely overlapping set of data samples, to collaboratively train a global model. The quality of data owners' local features affects the performance of the VFL model, which makes feature selection vitally important. However, existing feature selection methods for VFL either assume the availability of prior knowledge on the number of noisy features or prior knowledge on the post-training threshold of useful features to be selected, making them unsuitable for practical applications. To bridge this gap, we propose the Federated Stochastic Dual-Gate based Feature Selection (FedSDG-FS) approach. It consists of a Gaussian stochastic dual-gate to efficiently approximate the probability of a feature being selected. FedSDG-FS further designs a local embedding perturbation approach to achieve differential privacy for local training data. To reduce overhead, we propose a feature importance initialization method based on Gini impurity, which can accomplish its goals with only two parameter transmissions between the server and the clients. The enhanced version, FedSDG-FS++, protects the privacy for both the clients' training data and the server's labels through Partially Homomorphic Encryption (PHE) without relying on a trusted third-party. Theoretically, we analyze the convergence rate, privacy guarantees and security analysis of our methods. Extensive experiments on both synthetic and real-world datasets show that FedSDG-FS and FedSDG-FS++ significantly outperform existing approaches in terms of achieving more accurate selection of high-quality features as well as improving VFL performance in a privacy-preserving manner.

Index Terms—Vertical federated learning, feature selection, differential privacy, partially homomorphic encryption

1 Introduction

Obtaining a large amount of high-quality training data is crucial for building machine learning (ML) models in artificial intelligence (AI) applications. It is not only expensive (especially for data with high dimensions)e, but is also challenging due to privacy concerns precluding direct data sharing in many fields (*e.g.*, healthcare, finance). Federated learning (FL) [1], [2], [3], [4], [5], [6] is an emerging collaborative machine learning paradigm which enables multiple data owners (a.k.a., FL clients) to jointly train a model by iteratively exchanging model parameters with an FL server without exposing local data. It has been widely adopted in application such as safety monitoring [7], smart healthcare [8] and industrial fault detection [9]. FL can be broadly divided into two categories based on the distribution of local data [3]: 1) horizontal federated learning (HFL), and 2) vertical federated learning (VFL).

Under HFL [10], [11], [12], data owners' local datasets have little overlap in the sample space but large overlaps in the feature space. In contrast, under VFL [2], [13], [14], data owners' local datasets have large overlaps in the sample space but little overlap in the feature space. VFL scenarios often arise in applications in which companies from different business sectors collaborate to train a model [15], [16] (*e.g.*, an e-commerce company, a bank and a ride-sharing company could jointly build a model to identify potential financial fraudsters based on their unique perspectives on customer behaviour patterns through VFL). The quality of data owners' local features determines the effectiveness of their local models, which in turn, significantly affects the performance of the global VFL model. In practical applications, data owners often possess noisy features that are irrelevant to the learning task, or a large number of redundant features which negatively impact global model performance [17].

To improve the performance of VFL systems and reduce the cost of data acquisition, in this work, we focus on filtering noisy features and selecting important features. There are a number of feature selection methods for achieving high-performance models under centralized ML settings [18], [19], [20], while few work focused on VFL [21]. Feature selection methods for centralized ML can be divided into three categories. 1) Filter methods attempt to remove irrelevant features prior to learning a model. They calculate per-feature relevance scores based on statistical measures, *e.g.*, Gini impurity and mutual information [20], [22], [23]. 2) Wrapper methods search for the optimal feature subset in large search spaces, and thus, become computationally expensive, especially in the context of deep neural networks [24], [25]. 3) Embedded methods try to remove this burden by selecting the subset of important features while simultaneously learn the model [19], [26], [27].

In VFL, there are only three works on feature selection

- *Anran Li, Hongyi Peng, Luu Anh Tuan and Han Yu are with the school of Computer Science and Engineering of Nanyang Technological University, Singapore. E-mail: {anran.li, anhtuan.luu, han.yu}@ntu.edu.sg, hongyi001@e.ntu.edu.sg.*
- *Jiahui Huang, Lan Zhang and Xiang-Yang Li are with the school of Computer Science and Technology of University of Science and Technology of China, Hefei, China. E-mail: hjh233@mail.ustc.edu.cn, {zhanglan, xiangyangli}@ustc.edu.cn.*
- *Ju Jia is with the school of Cyber Science and Engineering, Southeast University, Nanjing 210096, China. E-mail: jiajuhb@gmail.com.*

(FS) [21], [28]. In [21], FS is performed with the filter method based on secure multi-party computation, while in [28], [29] the embedded method combined the auto-encoder with l_2 constraints on feature weights is used for FS. However, these works either assume prior knowledge on the number of noisy features [21] and the post-training threshold of useful features to be selected [28], or requires fine parameter tuning (*e.g.*, regularization parameters, number of pre-training epochs). These assumptions make them unsuitable for practical VFL applications. The problem of feature selection in VFL settings remains open. To enable feature selection to be performed in VFL settings, the following key research questions need to be addressed.

1) How to accurately identify noisy features, and select a small number of important features to train an optimal global VFL model? Existing feature selection methods for centralized ML require direct access to training samples, the training process and the labels simultaneously, which is not permitted in VFL. Besides, in those VFL works that try to protect the privacy of local data, intermediate parameters are transmitted in ciphertexts during VFL training [30], [31], which further increases the difficulty of feature selection.

2) How to conduct feature selection efficiently in VFL settings? Existing embedded feature selection methods require a large number of training iterations to select features, especially for high-dimensional data [19], [26]. Directly applying them in VFL will incur significant computation and communication overhead since each training round involves multiple privacy preservation operations, *e.g.*, encryption/decryption, and intermediate parameter transfers.

3) How to provide well trade-offs between privacy protection and system efficiency? In this work, we try to protect both the clients' training data and the server's labels. Protecting the private labels is necessary since the labels often contain highly sensitive information, *e.g.*, what a user has purchased in online advertising or whether a user has a disease or not in disease prediction. However, it is difficult to directly apply the partially homomorphic encryption (PHE) algorithm, *e.g.*, Paillier, to feature selection for vertical neural networks (NNs) since the nonlinear polynomials are not supported. Besides, different VFL systems have different privacy preservation requirements, and directly applying PHE techniques in all scenarios would incur substantial costs. Therefore, an adaptive privacy preservation method according to different privacy types to achieve a good trade-off between privacy protection and system efficiency is urgently needed.

To address the aforementioned questions and the limitations of existing works [17], [21], we propose Federated Stochastic Dual-Gate based Feature Selection (FedSDG-FS) and FedSDG-FS++ approaches. Aiming for adaptivity to systems with different privacy preferences, FedSDG-FS and FedSDG-FS++ protect the privacy of training data and the privacy of both the training data and labels, respectively. They are all embedded feature selection approaches consisting of a feature importance initialization module and a private important feature selection module. Our main contributions are summarized as follows.

- We propose FedSDG-FS and FedSDG-FS++ to accomplish both accurate feature selection and high-performance VFL model construction while adaptively satisfying various privacy preservation requirements.
- We propose the stochastic dual-gate and Gini impurity-based feature importance initialization method for FedSDG-FS and FedSDG-FS++ to ensure that the selected features are relevant to the context of the model. The stochastic dual-gate can efficiently approximate the probability of a feature and a embedding vector being selected, and can reduce the sizes of transmitted embedding vectors, thereby saving communication costs. The Gini impurity-based feature importance initialization method enables the global model to quickly filter out noisy features and select important ones, thus speeding up model training.
- We further design a local embedding perturbation approach for FedSDG-FS to achieve differential privacy for local training data. To protect the privacy of both the client's training data and the server's labels, FedSDG-FS++ leverages a secure feature selection approach based on PHE and the randomized noise mechanism. In these ways, FedSDG-FS and FedSDG-FS++ privately determine the selected features and produces an optimal global model with higher accuracy and fast convergence.

We evaluate FedSDG-FS and FedSDG-FS++ via extensive experiments using nine datasets including tabular data, images, texts and audios on a VFL system. The results show that they significantly outperform existing approaches in terms of achieving accurate and private selection of high-quality features to build high-performance VFL models. Taking `MADELON` dataset as an instance, the average test accuracy of FedSDG-FS is 27.0% higher than that of the best performing baseline with 47% fewer features required, and only half the communication cost.

2 Related Works & Preliminaries

Feature selection plays an important role in machine learning tasks. There are a number of feature selection methods proposed for centralized machine learning settings [18], [19], [20], while few works deal with feature selection in VFL [21]. In this section, we first present the existing work of feature selection both in centralized learning and FL. Then, we introduce the preliminaries of Gini impurity for filter-based feature selection.

2.1 Feature Selection in Centralized Learning

Feature selection methods in centralized learning settings can be divided into three categories: 1) filter methods, 2) wrapper methods, and 3) embedded methods.

1) Filter Methods: These methods attempt to remove irrelevant features prior to learning a model. A typical filter method consists of two steps, where the first step ranks features based on certain criteria, and in the second step, the features with highest rankings are chosen to induce classification models. Various performance criteria have been proposed for filter-based feature selection, *e.g.*, Gini impurity [18], Fisher score [20] and mutual information [22].

TABLE 1
Important notations.

Notation	Description
$\{U_1, \cdots, U_M\}$	a set of N samples $\{x_n, y_n\}_{n=1}^N$ owned by M clients;
$f_{m,j}$	the j-th feature of client m;
M, N	the total number of clients and samples;
$x_{n,m}$	the m-th block of the n-th sample vector x_n;
y_n	the n-th label among c class labels;
h_m	local embedding function of client m;
$h_{n,m}$	the local embedding of sample $x_{n,m}$;
$d_m, \underline{d_m}$	dimensions of sample $x_{n,m}$ and embedding $h_{n,m}$;
$\boldsymbol{\theta}, \theta_m, w_m, \theta_0$	parameters of the global model, the client m's local model, the interactive layer of client m and the top model;
$L(\cdot;\cdot), L_n$	the loss function and the loss of sample $x_{n,m}$;
s_m, q_m	the vectors of indicator variables;
μ_m, ω_m	clipped Gaussian variable parameters;

2) Wrapper Methods: Since filter methods select features independent of any particular models, they totally ignore the impacts of the selected subset of features on the performance of the induction algorithm [32], [33], which results in inaccurate feature selection for specific models. The optimal feature subset should depend on the specific biases and heuristics of the induction algorithm. Based on this assumption, wrapper methods leverage the outcomes of a model to determine the importance of each feature. They attempt to select a subset of features which can achieve the best prediction performance. As the number of subsets can be very large in the context of deep neural networks, and a model need to be recomputed for each subset, wrapper methods are generally computationally expensive [24], [25].

3) Embedded Methods: These methods aim to select a subset of relevant features, while simultaneously learning the model [19], [26], [27]. Embedded methods have the advantages of 1) wrapper methods - they consider the contextual information of the classification model and 2) filter methods - they are far less computationally intensive than wrapper methods. The least absolute shrinkage and selection operator [27] is a well-known embedded feature selection method, whose objective is to minimize the loss while enforcing an l_1 constraint on the weights of the features. However, embedded methods with regularized objective suffers from shrinkage of the model parameters, and cannot sparse the input layer [26], [34]. Another recently proposed method [19] uses a continuously relaxed Bernoulli variable to conduct feature selection based on stochastic gates. However, it requires a large number of parameters to be trained in the first layer of the model, resulting in overfitting to the training data, especially for deep neural networks with high-dimensional data or when there are only a limited number of training samples available.

Since these methods are designed for centralized learning scenarios in which all training data are accessible, such approaches are not applicable to VFL which demands data privacy protection for both local clients and the server. In addition, they are also not optimized to reduce communication or computation costs when the volume of training data is large, which make them not applicable in FL scenarios where clients are resource-constrained.

2.2 Feature Selection in VFL

In VFL, there are only a few existing works on feature selection and feature importance evaluation. SFFS [21] conducts feature selection with the filter method based on secure multi-party computation. However, since it performs VFL feature selection out of the context of the learning task, it can lead to inaccurate feature selection. Besides, it assumes that the number of noisy features is known in advance, and that there is a trusted third party for performing feature selection. These assumptions are unrealistic in practice. Further, it incur large communication overhead since a massive amount of parameters are transmitted between participants and the trusted third party. Other methods [28], [29] leverage the embedded method through combining the auto-encoder with l_2 constraints on feature weights. However, [28] suffers from the model parameter shrinkage [26] and requires post-training threshold setting to determine the selected features [17], while [29] requires fine parameter tuning (*e.g.*, number of pre-training epochs) and is vulnerable to reconstruction attacks and label leakage. The work [35] proposes two key factors affecting VFL performance - feature importance and feature correlation, and propose evaluation metrics and dataset splitting methods to improve model performance. However, it requires direct access to data features which breaches the privacy requirements of VFL. To this end, the proposed FedSDG-FS approach addresses these limitations of the state of the art.

2.3 Differential Privacy

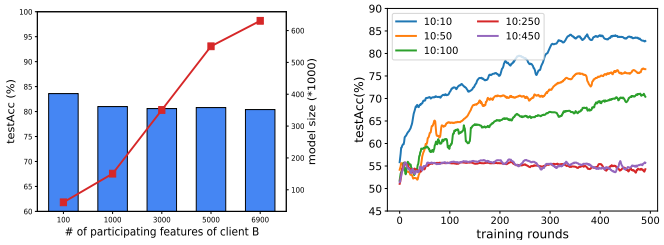
Definition 1 ((ϵ, δ)-DP [36])**.** *A randomized mechanism $\mathcal{M}: \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ϵ, δ)-DP if for any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$ and for any two adjacent inputs $d, d' \in \mathcal{D}$,*

$$\Pr[\mathcal{M}(d) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{M}(d') \in \mathcal{S}] + \delta. \tag{1}$$

Instead of the original definition of ϵ-DP, we adopt the variant [37] which allows for the possibility that plain ϵ-differential privacy is broken with probability δ. A common paradigm for approximating a real-valued function $f: d \to \mathbb{R}$ with a differentially private mechanism is via additive noise calibrated to f's *sensitivity* $\mathcal{S}_f$, which is defined as $\mathcal{S}_f = \max_{d,d'} |f(d) - f(d')|$. For instance, the Gaussian noise mechanism is defined as:

$$\mathcal{M}(d) = f(d) + \mathcal{N}(0, \mathcal{S}_f^2 \cdot \sigma^2) \tag{2}$$

where $\mathcal{N}(0, \mathcal{S}_f^2 \cdot \sigma^2)$ is the Gaussian distribution with mean 0 and standard deviation $\mathcal{S}_f \sigma$. In [38], a DP stochastic gradient decent algorithm (DP-SGD) has been proposed. It is similar to mini-batch gradient decent with the gradient averaging process being approximated by a Gaussian mechanism (GM). The basic idea for designing a differentially private additive-noise mechanism that implements a given functionality consists of the following steps: 1) approximating the functionality by a sequential composition of bounded-sensitivity functions; 2) choosing parameters of additive noise; and 3) performing privacy analysis of the resulting mechanism. Following this approach, we propose a differentially private feature selection method to prevent the transmission parameters (*e.g.*, local embedding vectors) from leaking clients' data information during the training and selection procedure.



(a) Accuracy and model size vs. number of participating features

(b) Accuracy vs. number of participating features

Fig. 1. Test accuracy and model size of vertical neural networks training using (a) the dataset ARECENE with redundant features; (b) the dataset MADELON with noisy features.

3 Problem Description

3.1 Basic Setup of VFL

There are two types of entities involved in VFL: a server S and M clients $\mathcal{M} := \{1, 2, \cdots, M\}$. A dataset $U = \{U_1, \cdots, U_M\}$ of N samples, $\{X, Y\} := \{x_n, y_n\}_{n=1}^N$, is maintained by the M clients. Let $[N] = \{1, 2, \cdots, N\}$. Each client m is associated with a unique set of features $\{f_{m,1}, \cdots, f_{m,d_m}\}$, and owns sample $x_{n,m} \in \mathbb{R}^{d_m}, n \in [N]$, where $x_{n,m}$ is the m-th block of the n-th sample vector $x_n := \left[x_{n,1}^\top, x_{n,2}^\top, \cdots, x_{n,M}^\top\right]^\top$. Suppose there are C possible class labels, the n-th label $y_n \in [C]$ is stored by server S. Typically, a data owner, which holds both the feature and the class labels, can act as the "FL server". It is referred to as the active party. Others which hold only features are referred to as the passive parties.

Each client m learns a local embedding h_m parameterized by $\theta_m \in \Theta$ that maps a high-dimensional vector $x_{n,m} \in \mathbb{R}^{d_m}$ into a low-dimensional one $h_{n,m} := h_m(\theta_m; x_{n,m}) \in \mathbb{R}^{\underline{d_m}}$ with $\underline{d_m} \ll d_m$. The server S learns the prediction $\hat{y}_n$ through the the top model θ_0 which is parameterized by $\theta_0 := \{w_1, \cdots, w_M, \alpha_0\} \in \Theta$, $w_m \in \mathbb{R}^{\underline{d_m}}$, $m \in [M]$, where $\{w_1, \cdots, w_M\}$ are parameters of the interactive layer which concatenates embedding vectors $h_{n,1}, \cdots, h_{n,M}$ in a weighted manner and α_0 denotes the parameters of the succeeding layers of the top model connected to the interactive layer. Generally, the objective of VFL is to minimize,

$$\begin{aligned} R(\boldsymbol{\theta}) &:= \mathbb{E}_{X,Y} L(h(\theta_0, h_{n,1}, \cdots, h_{n,M}); y_n) \\ &\text{with } h_{n,m} := h_m(\theta_m; x_{n,m}), m \in [M] \end{aligned} \tag{3}$$

where $\boldsymbol{\theta} := \{\theta_i\}_{i=0}^M$ denotes the global model, which consists of M local models $\theta_1, \cdots, \theta_M$ and the top model θ_0, and $L(\cdot;\cdot)$ is the loss function. This problem can be solved via iterative stochastic optimization. In the t-th iteration, the server receives embedding vectors $\{h_{n,m}^t\}_{m=1}^M$ from M clients. It then calculates and sends the gradients of the loss w.r.t. $h_{n,m}^t$ to all clients. Upon receiving the gradients, client m updates the local model to obtain θ_m^{t+1}. Then, client m randomly selects a datum $x_{n,m}$, calculates $h_{n,m}^{t+1}$ using θ_m^{t+1}, and sends it to the server. This process is repeated until the global model converges (*i.e.*, a convergence criterion is met). To ensure that neither data nor labels can be obtained or inferred by any other party, the above iterative training must be conducted in a privacy-preserving manner.

3.2 Motivating Analyses

Firstly, we conduct data driven analysis to demonstrate the necessity of feature selection in VFL. We illustrate this from two aspects: 1) many clients possess a large number of redundant features, which results in a low quality and very complex global model; and 2) some clients possess noisy or task irrelevant features which reduces global model performance. Specifically, we use datasets ARCENE [39] and MADELON [40] as training data. ARCENE contains 2,400 instances with 7,000 informative but redundant features. MADELON contains 4,400 instances with 5 informative features and 480 noisy features. We employ two clients, A and B, and a server to jointly train neural networks [30], *i.e.*, VFLNN–ARCENE and VFLNN–MADELON, based on these two datasets via VFL, and evaluate the test accuracy of the global model updated by by aggregating local bottom models of clients and the top model of the server.

To illustrate aspect 1), we assign different numbers of features of ARCENE to client B, while assigning 100 fixed features to client A to train VFLNN–ARCENE. The results in Fig 1(a) show that, as the number of redundant features increases, the test accuracy of the global model decreases slightly, while the model size grows rapidly. To illustrate aspect 2), we assign different numbers of noisy features from the MADELON dataset to client B, while assigning 10 fixed features to client A to train VFLNN–MADELON. The results are shown in Fig. 1(b), where $10 : k$ indicates that, A owns 10 features (*i.e.*, 3 informative features and 7 noisy features), and B owns k features (*i.e.*, 2 informative features and $(k-2)$ noisy features). It shows that as the number of noisy features increases, the test accuracy of the global VFL model decreases significantly. These results show that an efficient and privacy-preserving feature selection method is urgently needed for VFL.

3.3 Problem Formulation

In a typical VFL system, under the coordination of the server S, all participants jointly train a global model by transferring their local embedding vectors trained using their local datasets. We consider a practical situation that some clients possess a large number of noisy features or redundant features. This may result in a low-performance and extremely complex global model. Specifically, we can divide all features into qualified important features and negatively influential features, *e.g.*, noisy features or redundant features, by their effects to the objective of the global model. A desired VFL framework should enable all participants to jointly train a simple global model with a small number of important features, while eliminating negatively influential ones. The goal of feature selection in VFL is to simultaneously select a subset of features, and construct a global model $\hat{\boldsymbol{\theta}}$ with the objective by minimizing,

$$\begin{aligned} R(\boldsymbol{\theta}, s) &:= \mathbb{E}_{X,Y} L(h(\theta_0, h_{n,1}, \cdots, h_{n,M}); y_n) \\ &\text{with } h_{n,m} := h_m(\theta_m; x_{n,m} \odot s_m), m \in [M], \end{aligned} \tag{4}$$

where $s_m = \{0,1\}^{d_m}$ is the vector of indicator variables, and $s_{m,i}, i \in [d_m]$ are Bernoulli variables which indicate whether or not the i-th feature of client m is selected.

Security Assumptions: We assume that all participants are semi-honest. They follow the protocol of VFL and feature

selection without tampering with it and they do not collude with one another. Nevertheless, they are curious about other's private information and will try to infer as much as possible from the information received from the other participants. The semi-honest assumption is reasonable in our context since all participants have an incentive to learn a high-performance VFL model.

3.4 Privacy Threats

In this work, we consider two types of privacy threats, which expose the information about the training data of clients and the labels owned by the server.

•**Training Data.** The training data shall not be accessed by or exposed to any party other than the original owners. Besides, the transmitted intermediate parameters (*e.g.*, local embedding vectors) generated during forward propagation might leak private information (*e.g.*, data distributions) about the training data. Privacy-preserving techniques shall protect these aspects of training data.

•**Labels.** During backward propagation, the label information might be leaked from the server through transmitted intermediate parameters (*e.g.*, gradients of the training loss *w.r.t.* the interactive layer weights) [41], [42]. Thus, these parameters shall be protected.

3.5 Design Goals

We aim to design a VFL model training framework to support collaborative feature selection and model construction to achieve the following goals.

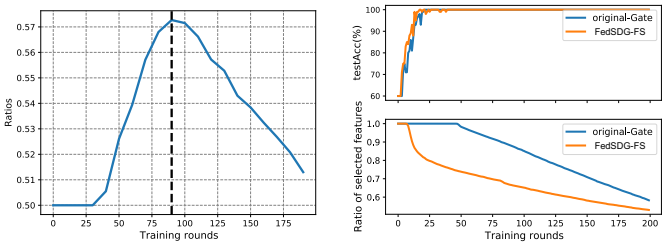
1) Effective Selection: The framework shall accurately select important features which have large positive effects on the global model performance, and exclude negatively influential features from training with the aim of improving model performance in terms of convergence speed and inference accuracy.

2) Privacy Preservation: For the above privacy threats, we aim to achieve two levels of privacy preservation. (1) **Level-1 privacy** protects transmission parameters (*e.g.*, local embedding vectors) from leaking clients' training data information during model training and feature selection. (2) **Level-2 privacy** protects both the clients' training data and the server's private labels from being exposed to or inferred by any party through the transmission parameters other than original owners.

3) Reducing Overhead: Considering the resource constraints of edge and mobile devices, the selection and updating process shall not incur high computation and communication cost.

4 The Proposed Approach

In this section, we first illustrate our proposed key techniques to enable feature selection to be performed jointly with model training under VFL settings. Then, we present the system design motivation and the system architecture. Finally, we present the details of the proposed approaches, which include feature importance initialization, DP-based important feature selection for FedSDG-FS and secure important feature selection for FedSDG-FS++.



(a) Ratio of the same selected features by stochastic gate and Gini impurity

(b) Comparison of the original gate method and FedSDG-FS

Fig. 2. Example motivation of FedSDG-FS design. The vertical neural network is trained on the dataset MADELON.

4.1 Stochastic Dual-Gates for VFL

To achieve accurate feature selection during model training, we need to dynamically calculate the influence of features on the global model, and increase the selection probabilities of features with high influence. In VFL, local embedding vectors are transferred to the server, where the size of them affects the communication cost. To reduce communication overhead in feature selection, we first introduce stochastic dual-gates for VFL to efficiently approximate the probabilities of features and dimensions of embedding vectors being selected. We re-express Eq. (4) into minimizing the l_0 constrained risk:

$$R(\boldsymbol{\theta}, s, q) := \mathbb{E}_{X,Y} L(h(\theta_0, g_{n,1}, \cdots, g_{n,M}); y_n) + \lambda \sum_m \left(|s_m|_0 + |q_m|_0\right) \tag{5}$$

where $h_{n,m} := h_m(\theta_m; x_{n,m} \odot s_m)$, $g_{n,m} = h_{n,m} \odot q_m$, $q_m = \{0,1\}^{\underline{d_m}}$ is the vector of indicator variables, where $q_{m,i}, i \in [d_m]$ are Bernoulli variables and indicate whether or not the i-th dimension of embedding $h_{n,m}$ is selected for global model training, and λ is a weighting factor for the regularization. We use l_0 norm to penalize the number of non-zero entries in the vectors s_m, q_m, and encourage the sparsity in the final estimates. Notice that l_0 norm induces no shrinkage on the actual values of the parameters, which is in contrast to l_1 regularization [27].

To solve this problem is not straightforward, since the optimization of hard feature selection with binary masks suffers from high variance. To this end, we propose a secure Gaussian-based continuous relaxation for the Bernoulli variables for VFL. We approximate each element of s_m, q_m to clipped Gaussian random variables parameterized by μ_m, ω_m as $s_{m,i} = \max(0, \min(1, \mu_{m,i} + \rho_{m,i}))$, $q_{m,j} = \max(0, \min(1, \omega_{m,j} + \gamma_{m,j}))$, where $\rho_{m,i}, \gamma_{m,j}$ are drawn from $\mathcal{N}(0, \sigma^2)$, and $\mu_{m,i}, \omega_{m,j}$ can be learned during VFL training. Under the continuous relaxation, the regularization term in Eq. (5) is simply the sum of the probabilities that $\sum_{i\in[d_m]} P(s_{m,i} > 0) + \sum_{j\in[\underline{d_m}]} P(q_{m,j} > 0)$, and can be calculated by $\sum_{i\in[d_m]} \Phi(\frac{\mu_{m,i}}{\sigma}) + \sum_{j\in[\underline{d_m}]} \Phi(\frac{\omega_{m,j}}{\sigma})$, where $\Phi(\cdot)$ is the cumulative distribution function of the standard Gaussian distribution. By leveraging the continuous distribution, we can thus transform Eq. (5) into the following:

$$R(\boldsymbol{\theta}, \mu, \omega) := \mathbb{E}_{X,Y} L(h(\theta_0, g_{n,1}, \cdots, g_{n,M}); y_n) + \lambda \left(\sum_{m,i} \Phi\left(\frac{\mu_{m,i}}{\sigma}\right) + \sum_{m,j} \Phi\left(\frac{\omega_{m,j}}{\sigma}\right) \right). \tag{6}$$

To optimize the objective of Eq. (6), we first differentiate it with respect to μ_m, ω_m. However, since the loss L of the global model is calculated and stored at the server S, client m cannot directly access it. Thus, client m performs the differentiation using chain rules [43] based on the Monte Carlo sampling gradient estimator, *e.g.*, for μ_m:

$$\frac{1}{Q}\sum_{i\in[Q]}\left[\frac{\partial L_n}{\partial g_{n,m}}\cdot\frac{\partial g_{n,m}}{\partial s_m}\cdot\frac{\partial s_{m,i}}{\partial \mu_{m,i}}\right]+\lambda\frac{\partial}{\partial \mu_m}\Phi\left(\frac{\mu_m}{\sigma}\right) \quad (7)$$

where Q is the number of Monte Carlo samples. The calculation of gradient of estimator for ω_m is similar to Eq. (7). Thus, we can update μ_m, ω_m via stochastic gradient descent.

4.2 System Design Motivations

There are many challenges in solving this optimization problem. Firstly, updating the parameters and conducting the above operations require access to all local training samples or training processes, which are hidden from any third party including the server in FL. Thus, how to accurately calculate them in a privacy-preserving manner is challenging. Secondly, it would require a large number of parameters to be trained (*e.g.*, μ_m, $m \in [M]$) by directly applying the stochastic dual-gates to the clients' inputs and local embedding vectors, which slows down the convergence of the global model and incurs significant computation and communication overhead, especially for high-dimension features.

To address these challenges, we propose efficient and privacy-preserving feature selection frameworks, FedSDG-FS and FedSDG-FS++, which commonly adopts the Gini impurity-based important feature initialization to facilitate feature selection and reduce computation and communication cost. Then, important features and significant local embeddings can be selected by the proposed stochastic dual gates, enhanced with DP and local perturbations in FedSDG-FS, and enhanced with PHE and the randomized noisy mechanism in FedSDG-FS++ to achieve privacy preservation. We make the following three empirical observations to illustrate the motivation of the importance initialization. i) As illustrated in Fig. 2(a), there can be a large ratio of the same features being selected by the Gini impurity [18] and by the stochastic gates in some training rounds. ii) Gini impurity initialization can speed up feature selection (Fig. 2(b)). iii) The reason that Gini impurity cannot be directly used for feature selection is that it cannot take into account the specific VFL models and has no prior knowledge of the number of important features to select. The feature importance initialization step can be accomplished through two parameter transmissions with two encryption/decryption operations on the server based on Gini impurity and PHE, which significantly improves efficiency and privacy preservation. In this way, we can achieve efficient and private feature selection while constructing the global VFL model.

4.3 System Overview

FedSDG-FS and FedSDG-FS++ consist of two modules (as shown in Fig. 3):

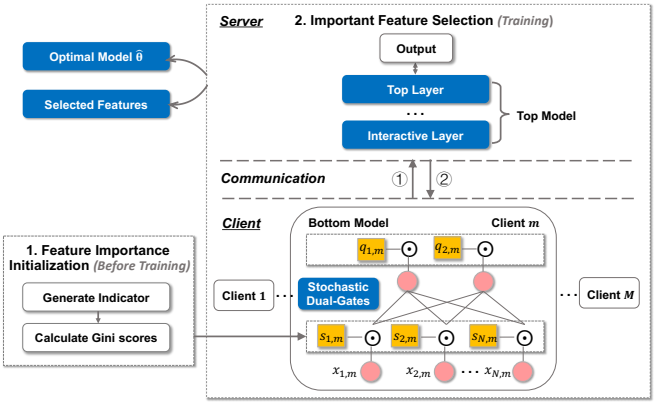


Fig. 3. System overview of FedSDG-FS. ① Send encrypted embeddings, ② send encrypted gradients.

1) Feature Importance Initialization before Training. To save feature selection costs, local clients first securely initialize feature importance based on Gini impurity and PHE, in cooperation with the server prior to the commencement of model training.

2) Important Feature Selection during Training. After feature importance initialization, the server coordinates clients to select important features, while training the VFL model for improved performance. Specifically, in FedSDG-FS, to achieve Level-1 privacy, we propose a DP-based feature selection approach based on local perturbation on embedding functions. While in FedSDG-FS++, to achieve Level-2 privacy, we propose a secure feature selection approach which includes forward propagation for secure feature selection, and backward propagation for secure feature selection, based on the proposed stochastic dual-gate, PHE and the randomized noise mechanism. In this way, FedSDG-FS and FedSDG-FS++ determine the selected features and produces optimal global models $\hat{\theta}$ with higher accuracy and fast convergence.

4.4 FedSDG-FS

4.4.1 *Feature Importance Initialization*

As illustrated in Section 3.1 that M clients have a set $U = \{U_1, \cdots, U_M\}$ of N samples, for client m, if the j-th feature $f_{m,j}$ is a discrete feature that can assume b values, then it induces a partition $U_{m,1} \cup \cdots \cup U_{m,c}$ of the set U_m in which $U_{m,i}$ is the set of instances with the i-th value for $f_{m,j}$. The Gini impurity of $U_{m,i}$ is defined as $G(U_{m,i}) = 1 - \sum_{k\in[C]} p_{m,k}^2$, where $p_{m,k}$ is the probability of a randomly selected instance from $U_{m,i}$ that belongs to the k-th class. The Gini score of feature $f_{m,j}$ is calculated as $G(f_{m,j}) = \sum_{i\in[c]} \frac{|U_{m,i}|}{|U_m|} \cdot G(U_{m,i})$, where $G(f_{m,j})$ measures the likelihood of a randomly selected instance being misclassified. If $f_{m,j}$ is a feature with continuous values, then $G(f_{m,j})$ is defined as the weighted average of the Gini impurities of a set of discrete feature values. We use the Paillier as the PHE method which supports homomorphic addition of two ciphertexts and homomorphic multiplication between a plaintext and a ciphertext. The calculation of $p_{m,k}$ requires collaboration between client m and the server. Thus, we design an efficient and secure collaborative calculation protocol, which is shown in Algorithm 1.

Algorithm 1: Feature Importance Initialization

Input : Server S, clients m
Output: Initialized feature importance
1 **Server** S
2 Generate an indicator matrix A, $[\![A]\!] \leftarrow Enc(A)$
3 Send $[\![A]\!]$ to all clients
4 **Client** m
5 Induce a partition $U_{m,1} \cup U_{m,2} \cup \cdots \cup U_{m,b}$ of U_m
6 Calculate $[\![p_{m,k}]\!] \leftarrow \sum_{a \in I(U_{m,i})} [\![A]\!]_{a,k}/|U_{m,i}|$
7 Calculate $[\![p_{m,k}]\!]^2$ with the protocol in [44]
8 $[\![G(U_{m,i})]\!] \leftarrow 1 - \sum_{k \in [c]} [\![p_{m,k}]\!]^2$
9 $[\![G(f_{m,j})]\!] \leftarrow \sum_{i=1}^{c} \frac{|U_{m,i}|}{|U_m|} \cdot [\![G(U_{m,i})]\!]$
10 Send $[\![G(f_{m,j})]\!], j \in [d_m]$ to the server
11 **Server** S
12 $G(f_{m,j}) \leftarrow Dec([\![G(f_{m,j})]\!])$
13 Send $G(f_{m,j}), j \in [d_m]$ to client m
14 **Client** m
15 Initialize $\mu_{m,j} \propto \frac{1}{G(f_{m,j})}$
16 **Return** feature importance initialization $\mu_{m,j}, j \in [d_m]$

Specifically, the server first generates an indicator matrix A with a size of $N \times C$, where $A_{n,k} = 1$ denotes the category of the n-th sample is k; otherwise, $A_{n,k} = 0$. Then, the probability $p_{m,k}$ of client m can be calculated as $p_{m,k} = \sum_{a \in I(U_{m,i})} A_{a,k}/|U_{m,i}|$, where $I(U_{m,i})$ indicates the index set of instances from $U_{m,i}$. To prevent the label information from being leaked, the server encrypts the matrix A, and sends $[\![A]\!]$ to all clients (Line 2-3). Then, client m calculates the probability $[\![p_{m,k}]\!] = \sum_{a \in I(U_{m,i})} [\![A]\!]_{a,k}/|U_{m,i}|$ and uses the protocol in [44] to compute the square of $[\![p_{m,k}]\!]$ as follows. Firstly, client m generates a random value r and computes $[\![u_{m,k}]\!] = [\![p_{m,k} + r]\!]$, such that $p_{m,k}^2$ equals $u_{m,k}^2 - 2u_{m,k} \cdot r + r^2$ and $[\![-2u_{m,k} \cdot r + r^2]\!]$ can be locally computed by the client (Line 5-10). Then, client m sends $[\![u_{m,k}]\!]$ to the server. The server decrypts it, computes and sends $[\![u_{m,k}^2]\!]$ to client m (Line 12-13). Finally, client m computes $[\![p^2]\!] = [\![u_{m,k}^2 - 2u_{m,k} \cdot r + r^2]\!]$. After calculating $[\![p^2]\!]$, client m calculates the Gini impurity $[\![G(f_{m,j})]\!]$ of feature $f_{m,j}$, and sends them to the server (Line 15-16). The server then decrypts them, and assigns larger initial importance values, *i.e.*, larger initial value of μ_m, to features with smaller Gini values. During this process, only the server learns the Gini scores of client m's features, while other parties learn nothing. It shows that the calculation of Gini impurity only involves two parameter transmissions between the server and the clients. The computation cost of each client is $O(1)$ operations due to multiple addition and multiplication operations, while the communication cost of each client is $O(size([\![A]\!]) + \frac{\sum_{m=1}^{M} d_m}{M} \cdot size([\![G(f_{m,j})]\!]))$ where $size([\![A]\!])$, $size([\![G(f_{m,j})]\!])$ denote sizes of $[\![A]\!]$ and $[\![G(f_{m,j})]\!]$, respectively.

4.4.2 Differentially Private Important Feature Selection

During the feature selection and updating process, clients keep sending the embedding vectors to the server, which might leak training data information [38], [45]. To this end, we propose a privacy-preserving feature selection framework based on the Gaussian DP mechanism [46] to enhance data confidentiality of VFL participants (with negligible training time), test accuracy in the feature selection and updating process.

Local Perturbation. As illustrated in Section 3.1, h_m denotes the local embedding function of client m with the parameter θ_m which embeds the input data $x_{n,m}$ into its output $h_{n,m} := h_m(\theta_m; x_{n,m} \odot s_m)$. When h_m is linear embedding, it is as simple as $h_m(\theta_m; x_{n,m}) = (x_{n,m} \odot s_m)^\top \theta_m$. For non-linear embeddings such as neural networks, $h_{n,m}$ can be represented as:

$$\begin{aligned} u_m^0 &= x_{n,m} \odot s_m \\ u_m^i &= \sigma_i(w_m^i u_{i-1} + b_m^i),\ i \in [I] \\ h_{n,m} &= u_m^I \end{aligned} \tag{8}$$

where $\sigma_i(\cdot)$ is a linear or non-linear function, w_m^i, b_m^i corresponds to the parameter θ_m^i of h_m (*e.g.*, $\theta_m^i = \{w_m^i, b_m^i\}^\top$) and θ_m^i is the i-th layer parameter of θ_m. Here, we use K_{σ_i} to denote the lipschitz constant of a function $\sigma_i(\cdot)$, and we assume that σ_i is K_{σ_i}-Lipschitz continuous. Specially, when h_m is linear, the composite form embedding corresponds to $I = 1, \sigma_1(w_m^1 x_{n,m} \odot s_m + b_m^1) = w_m^1 x_{n,m} \odot s_m + b_m^1$. We perturb the local embedding function h_m by adding a random neuron with output z_m^i at each layer i:

$$\widetilde{u}_m^i = \sigma_i(w_m^i u_{i-1} + b_m^i + z_m^i),\ i \in [I]. \tag{9}$$

$\widetilde{h}_{n,m} = u_m^I$, where $z_m^1, \cdots, z_m^I$ are independent random variables. We show h_m is smooth and enables DP with properly chosen distribution of $z_m^i, i \in [I]$. We set the distribution as $z_m^I \sim \mathcal{N}(0, \zeta_m^2)$, $z_m^i \sim \mathcal{U}[-\sqrt{3}\zeta_m^i, \sqrt{3}\zeta_m^i], i \in [I-1]$, where $\mathcal{N}(0, \zeta_m^2)$ denotes the Gaussian distribution with zero mean and variance ζ_m^2, and $\mathcal{U}[-\sqrt{3}\zeta_m^i, \sqrt{3}\zeta_m^i]$ denotes the uniform distribution over $[-\sqrt{3}\zeta_m^i, \sqrt{3}\zeta_m^i]$.

Enforcing Smoothness. After the embedding function h_m is perturbed, to guarantee the convergence of model training considering the feature selection, the objective function (Eq. (6)) should be smooth. Inspired by the randomized smoothing technique, we are able to smooth the objective function by taking expectation with respect to random neurons, which follows the fact that the smoothness of a function can be increased by convolving with proper distributions. Specifically, by adding a random neuron z_m^i, σ_i will be smoothed in expectation with respect to z_m^i, and the local embedding vector $\widetilde{h}_{n,m}$ is smooth by induction. Then the global objective R is smooth by taking expectation with respect to all the random neurons when the loss function $L(\cdot)$ is smooth w.r.t. the local embedding vector $h'_{n,m}$. Further, the perturbed loss is smooth w.r.t. the local model θ_m, and a large perturbation (large ζ_m^i or ζ_m) will lead to a smaller smoothness constant.

GDP-based Feature Selection. We now leverage the local embedding perturbation technique in the private information transmission and achieve privacy-preserving feature selection and model updating (see Algorithm 2). Client $m \in [M]$ randomly selects a private datum (or mini-batch) $x_{n,m}$, and calculates the indicator $s_{m,i}$ for each feature $f_{m,i}$, $i \in [d_m]$. Then, it perturbs the local embedding function h_m using Eq. (8) with the random neuron $z_m^i, i \in [I]$, and calculates the perturbed embedding vector $h'_{n,m}$ and the masked embedding $g'_{n,m}$ which is sent to the server.

Then, the server calculates the gradients of the loss w.r.t. the interactive layer weight $\frac{\partial L_n}{\partial w_m}$, and w.r.t. the succeeding layer weights $\frac{\partial L_n}{\partial \alpha_0}$, and updates its top model θ_0 (Line 13-14). It then calculates the gradient $\frac{\partial L_n}{\partial g'_{n,m}}$ of the loss w.r.t. the masked embedding $g'_{n,m}$, and sends $\frac{\partial L_n}{\partial g'_{n,m}}$ to client m. Upon receiving the gradient $\frac{\partial L_n}{\partial g'_{n,m}}$, client m updates the local model θ_m and the selection variables μ_m, ω_m (Line 17-19). This process is repeated until the global model converges. In this way, model update and feature selection can be accomplished simultaneously.

We further provide privacy guarantees and utility analyses for Algorithm 2. We first define notations and make some assumptions.

Definition 2. *A mechanism $\mathcal{M}$ is said to satisfy β-GDP if for all neighboring datasets D and D', we have*

$$H(\mathcal{M}(D), \mathcal{M}(D')) \leq H(\mathcal{N}(0,1), \mathcal{N}(\beta, 1)) \tag{10}$$

where the trade-off function $H(\cdot;\cdot)(d) = inf\{a_\phi : d_\phi \leq d\}$ are type I and type II errors given a threshold ϕ.

Intuitively, β-GDP guarantees that distinguishing two adjacent datasets via information revealed by $\mathcal{M}$ is at least as difficult as distinguishing the two distributions $\mathcal{N}(0,1)$ and $\mathcal{N}(\beta,1)$. Smaller β means less privacy loss. We leverage the moments accountant technique [38] to characterize the level of privacy of the GDP-based feature selection method.

Algorithm 2: GDP-based Private Feature Selection

Input : M clients with N samples $\{x_n, y_n\}_{n=1}^N$, $x_{n,m} \in \mathbb{R}^{d_m}$, $\zeta, \zeta_i, i \in [I-1]$

Output: Global model $\hat{\theta}$, indicator vector $\{s_m\}_{m=1}^M$

1 Initialize model $\theta_0 := \{\alpha_0, w_1, w_2, \cdots, w_M\}$, $\{\theta_i\}_{i=1}^M$, $\omega_m \in \mathbb{R}^{d_m}$, initialize μ_m with Algorithm 1

2 **Client** m, $m \in [M]$

3 Select datum $x_{n,m}, u_m^0 = x_{n,m} \odot s_m$

4 Sample $\rho_{m,i}, \gamma_{m,j} \sim \mathcal{N}(0, \sigma^2)$, $i \in [d_m], j \in [\underline{d_m}]$

5 Compute $s_{m,i} = \max(0, \min(1, \mu_{m,i} + \rho_{m,i}))$

6 $q_{m,j} = \max(0, \min(1, \omega_{m,j} + \gamma_{m,j}))$

7 $R_m = \sum_{i\in[d_m]} \Phi\big(\frac{\mu_{m,i}}{\sigma}\big) + \sum_{j\in[\underline{d_m}]} \Phi\big(\frac{\omega_{m,j}}{\sigma}\big)$

8 **for** $i = 1, \cdots, I-1$ **do**

9 $z_m^i \sim \mathcal{U}[-\sqrt{3}\zeta_m^i, \sqrt{3}\zeta_m^i]$

10 $\widetilde{u}_m^i = \sigma_i(w_m^i u_m^{i-1} + b_m^i + z_m^i)$

11 $z_m^I \sim \mathcal{N}(0, \zeta_m), \widetilde{h}_{n,m} = \widetilde{u}_m^I, \widetilde{g}_{n,m} = \widetilde{h}_{n,m} \odot q_m$

12 Send $\widetilde{g}_{n,m}$ to the server

13 **Server** S

14 Compute the gradients $\frac{\partial L_n}{\partial w_m}, m \in [M]$, $\frac{\partial L_n}{\partial \alpha_0}$

15 Update top model $\theta_0 = \{w_1, \cdots, w_M, \alpha_0\}$ as $w_m \leftarrow w_m - \eta_0 \frac{\partial L_n}{\partial w_m}, m \in [M]$, $\alpha_0 \leftarrow \alpha_0 - \eta_0 \nabla_{\alpha_0} L_n$

16 Compute the gradient $\frac{\partial L_n}{\partial g'_{n,m}}$, send it to client m

17 **Client** $m \in [M]$

18 Calculate $\frac{\partial L_n}{\partial \mu_m}, \frac{\partial L_n}{\partial \omega_m}, \frac{\partial L_n}{\partial \theta_m}$

19 Update $\mu_m \leftarrow \mu_m - \eta_m\big(\frac{\partial L_n}{\partial \mu_m} + \lambda \frac{\partial R_m}{\partial \mu_m}\big)$

20 $\omega_m \leftarrow \omega_m - \eta_m\big(\frac{\partial L_n}{\partial \mu_m} + \lambda \frac{\partial R_m}{\partial \omega_m}\big)$, $\theta_m \leftarrow \theta_m - \eta_m \frac{\partial L_n}{\partial \theta_m}$

21 **Return** the global model $\theta = \{\theta_m\}_{m=0}^M$.

Assumption 1. *The optimal loss is lower bounded $L^* > -\infty$. The gradient $\frac{\partial R(\theta)}{\partial \theta_0}$ is K-Lipschitz continuous, and $\frac{\partial R(\boldsymbol{\theta})}{\partial \theta_m}, m \in [M]$ is K_m-Lipschitz continuous.*

Theorem 1. *Under Assumption 1 and the assumptions that, for each embedding function h_m, if the activation functions $\sigma_i = \sigma, \forall i$, and $||w_m^i||$ is bounded, for client m, if we set the variance of the Gaussian random neuron at the I-th layer as $\zeta = \mathcal{O}(b_m\sqrt{e}/(\beta b))$, where b_m is the size of batch of client m, $b = \sum_{m=1}^M b_m$ is the size of the whole batch, and e is the number of queries (*i.e. *the number of data samples processed by h_m at client m), then the Algorithm 2 of FedSDG-FS satisfies β-GDP for the dataset U_m of client m.*

Theorem 1 illustrates the trade-off between model performance and local data privacy. Specifically, to increase data privacy, *i.e.*, decrease β in Eq. (10), we can increase the variance of random neurons. While as the variance of random neurons increases, the variance of the stochastic gradients of the loss w.r.t. top model θ_0, and w.r.t. local model $\theta_m, m \in [M]$ also increase, which will in turn lead to lower test accuracy or slower convergence. Thus, through Theory 1, we can set the proper value of β according to the requirements of model performance and local data privacy so as to achieve a good trade-off between them.

4.5 FedSDG-FS++

The feature importance initialization phase of FedSDG-FS++ is the same as FedSDG-FS (illustrated in Section 4.4.1), thus we directly present the secure important feature selection. We use the Paillier as the PHE method. The detailed processes of forward propagation and backward propagation are shown in Algorithm 3 and Algorithm 4.

1) Forward Propagation on Clients. Client m randomly selects a private datum (or mini-batch) $x_{n,m}$, and calculates the indicator $s_{m,i}$ for each feature $f_{m,i}$, $i \in [d_m]$. Then, it calculates the embedding vector $h_{n,m}$ using the local model θ_m and the masked embedding $g_{h,m} = h_{n,m} \odot q_m$, and encrypts it with PHE to obtain $[\![g_{n,m}]\!] = Enc(g_{n,m})$, which is sent to the server (Line 2-9 of Alg. 3).

2) Forward Propagation on the Server. After receiving the encrypted embedding $[\![g_{n,m}]\!]$, the server calculates the weighted vector $[\![z_{n,m}]\!] = [\![g_{n,m}]\!] \odot w_m$, and performs the forward propagation of the top model. Since the non-linear activation function on the top model cannot be calculated on the encrypted data, the weighted vector $[\![z_{n,m}]\!]$ should be sent back to client m for decryption. However, sending the weighted vector directly without any protection would leak the prediction to the client (*e.g.*, client m can use the activation prediction pair $(z_{n,m}, g_{n,m})$ to infer activation values and weights of the top model). To prevent this, the server adds random noises ϵ_s on $[\![z_{n,m}]\!]$, and sends $[\![z_{n,m} + \epsilon_s]\!]$ to client m. Then, client m decrypts the noisy weighted sum $[\![z_{n,m} + \epsilon_s]\!]$, and sends $z_{n,m} + \epsilon_s$ to the server (Line 11-14 of Alg. 3). Finally, the server removes the noise and computes the activation for the next layer. The process repeats until the final layer is reached. Another problem is that the server holds both w_m and $z_{n,m}$, and can easily infer $h_{n,m}$ via linear regression. To avoid this, the server should use the noisy weight $\widetilde{w}_m$ to calculate the weighted vector

$[\![\widetilde{z}_{n,m}]\!] \leftarrow [\![g_{n,m}]\!] \odot \widetilde{w}_m$, where $\widetilde{w}_m = w_m + \epsilon_{acc}$, ϵ_{acc} is generated by the client.

3) Backward Propagation on the Server. To update the global model, two gradients need to be computed first, the loss gradients w.r.t. the weight of the interactive layer $\frac{\partial L_n}{\partial w_m}$, and w.r.t. the embedding vector $\frac{\partial L_n}{\partial g_{n,m}}$. Since these two gradients are linear transformations of either $g_{n,m}$ or w_m, both the server and client m can derive what they want to acquire via regression. To this end, we design the following secure backward propagation method.

Specifically, the server first calculates the following gradients: $[\![\frac{\partial L_n}{\partial w_m}]\!]$, $\frac{\partial \widetilde{L}_n}{\partial g_{n,m}}$, $\frac{\partial L_n}{\partial \alpha_0}$. If the server updates $[\![w_m]\!]$ by $[\![w_m]\!] = w_m - \eta_m [\![\frac{\partial L_n}{\partial w_m}]\!]$, this would result in two encrypted quantities in calculating the weighted vector $z_{n,m} = [\![w_m]\!][\![g_{n,m}]\!]$, which is incompatible with PHE. To avoid this, the server needs to send $[\![\frac{\partial L_n}{\partial w_m}]\!]$ to client m, and receive the decrypted gradient $\frac{\partial L_n}{\partial w_m}$ back. However, sending $[\![\frac{\partial L_n}{\partial w_m}]\!]$ directly to client m would leak information about both parties, because the server holds $\frac{\partial L_n}{\partial z_{n,m}}$ and client m holds $h_{n,m}$. Thus, both the server and client m need to add random noises to the encrypted gradient of weights $\frac{\partial L_n}{\partial z_{n,m}}$ before sending them to the other party, and update the parameters (Line 3-10 of Alg. 4). Note that the noise ϵ_s generated by the server can be removed when the gradient $\frac{\partial \widetilde{L}_n}{\partial w_m}$ still contains noise, where $\frac{\partial \widetilde{L}_n}{\partial w_m} = \frac{\partial L_n}{\partial w_m} - \frac{\epsilon_m}{\eta_0}$. With $\frac{\partial \widetilde{L}_n}{\partial w_m}$, the server updates the weights as $\widetilde{w}_m^{t+1} = w_m^t - \eta(\frac{\partial L_n}{\partial w_m} - \frac{\epsilon_m}{\eta_0}) = w_m^{t+1} + \epsilon_m$.

It can be observed that the noise ϵ_m will accumulate in weights w_m in each iteration. If we take the accumulated noise as $\epsilon_{acc} = \sum_{m=1}^{M}\sum_{i=1}^{t}\epsilon_m^i$, the true weights used in forward and backward propagation should be $w_m^{t+1} = \widetilde{w}_m^{t+1} - \epsilon_{acc}$. To perform the correct forward operation, client m needs to remove the noise by subtracting $g_{n,m}\epsilon_{acc}$ from the noisy weighted vector $\widetilde{z}_{n,m}$. Similarly, the extra noise should be added to $\frac{\partial \widetilde{L}_n}{\partial g_{n,m}}$, and removed before backpropagation by client m. To achieve this, client m needs to send the encrypted noise $[\![\epsilon_{acc}]\!]$ to the server, and the server calculates the true gradient via $[\![\frac{\partial L_n}{\partial g_{n,m}}]\!] = \frac{\partial \widetilde{L}_n}{\partial g_{n,m}} - [\![\epsilon_{acc}]\!] \cdot \frac{\partial L_n}{\partial z_{n,m}}$, and sends the encrypted gradient $[\![\frac{\partial L_n}{\partial g_{n,m}}]\!]$ to the client m.

4) Backward Propagation on Clients. The client m first decrypts the gradient $[\![\frac{\partial L_n}{\partial g_{n,m}}]\!]$ received from the server. Then, it updates the local model θ_m and the variables μ_m, ω_m simultaneously (Line 16-20 of Alg. 4).

We further present discussions about the setting where clients have diverse computing and communication resources and results the problem of asynchrony and delays. For this scenario, we can add existing asynchronous and parallel optimization methods [13] on top of FedSDG-FS. Specifically, we describe the asynchronous FedSDG-FS in a high level as follows. During the learning and feature selection process, from the server side, it waits until receiv-

Algorithm 3: Forward Propagation for Secure Feature Selection

Input : M clients with N samples $\{x_n, y_n\}_{n=1}^N$, $x_{n,m} \in \mathbb{R}^{d_m}$
Output: Global model $\hat{\theta}$, indicator vector $\{s_m\}_{m=1}^M$

1 Initialize model $\theta_0 := \{\alpha_0, w_1, w_2, \cdots, w_M\}$, $\{\theta_i\}_{i=1}^M$, noise ϵ_{acc}, $\omega_m \in \mathbb{R}^{\underline{d_m}}$, initialize μ_m with Algorithm 1
2 **Client** m, $m \in [M]$
3 Select datum (or data mini-batch) $x_{n,m}$
4 Sample $\rho_{m,i}, \gamma_{m,j} \sim \mathcal{N}(0, \sigma^2)$, $i \in [d_m]$, $j \in [\underline{d_m}]$
5 Compute $s_{m,i} = \max(0, \min(1, \mu_{m,i} + \rho_{m,i}))$
6 $q_{m,j} = \max(0, \min(1, \omega_{m,j} + \gamma_{m,j}))$
7 $R_m = \sum_{i\in[d_m]} \Phi\big(\frac{\mu_{m,i}}{\sigma}\big) + \sum_{j\in[\underline{d_m}]} \Phi\big(\frac{\omega_{m,j}}{\sigma}\big)$
8 $h_{n,m} \leftarrow h_m(\theta_m; x_{n,m} \odot s_m)$, $g_{n,m} = h_{n,m} \odot q_m$
9 $[\![g_{n,m}]\!] \leftarrow Enc(g_{n,m})$, sends $[\![g_{n,m}]\!]$ to the server
10 **Server** S
11 Calculate the noisy weight $\widetilde{w}_m \leftarrow w_m + \epsilon_{acc}$
12 Compute $[\![\widetilde{z}_{n,m}]\!] \leftarrow [\![g_{n,m}]\!] \cdot \widetilde{w}_m$
13 Add random noise $[\![\widetilde{z}_{n,m} + \epsilon_s]\!] \leftarrow [\![\widetilde{z}_{n,m}]\!] + \epsilon_s$
14 Send $[\![\widetilde{z}_{n,m} + \epsilon_s]\!]$ to client m
15 **Client** m, $m \in [M]$
16 $\widetilde{z}_{n,m} + \epsilon_s \leftarrow Dec([\![\widetilde{z}_{n,m} + \epsilon_s]\!])$
17 Remove noise $z_{n,m} + \epsilon_s \leftarrow \widetilde{z}_{n,m} + \epsilon_s - \epsilon_{acc} g_{n,m}$
18 Send $z_{n,m} + \epsilon_s$ to the server
19 **Server** S
20 Remove noise $z_{n,m} \leftarrow z_{n,m} + \epsilon_s - \epsilon_s$
21 Compute $L_n \leftarrow L(h(\alpha_0, z_{n,1}, \cdots, z_{n,M}); y_n)$
22 **Return** the loss L_n

Algorithm 4: Backward Propagation for Secure Feature Selection

Input : Loss L_n on the server, target $\{y_n\}_{n=1}^N$, learning rates η_0, η_m
Output: Global model $\hat{\theta}$, indicator vector $s_m, q_m, m \in [M]$

1 **Server** S
2 Compute the gradients $[\![\frac{\partial L_n}{\partial w_m}]\!] \leftarrow \frac{\partial L_n}{\partial z_{n,m}} \cdot [\![g_{n,m}]\!]$, $\frac{\partial \widetilde{L}_n}{\partial g_{n,m}} \leftarrow \frac{\partial L_n}{\partial z_{n,m}} \cdot \widetilde{w}_m$, $\frac{\partial L_n}{\partial \alpha_0}$
3 Add noise $[\![\frac{\partial L_n}{\partial w_m} + \epsilon_s]\!] \leftarrow [\![\frac{\partial L_n}{\partial w_m}]\!] + \epsilon_s$
4 Send $[\![\frac{\partial L_n}{\partial w_m} + \epsilon_s]\!]$ to client m
5 **Client** $m \in [M]$
6 $\frac{\partial L_n}{\partial w_m} + \epsilon_s \leftarrow Dec([\![\frac{\partial L_n}{\partial w_m} + \epsilon_s]\!])$
7 Add noise $\frac{\partial \widetilde{L}_n}{\partial w_m} + \epsilon_s \leftarrow \frac{\partial L_n}{\partial w_m} + \epsilon_s - \frac{\epsilon_m}{\eta_0}$
8 Encrypt noise $[\![\epsilon_{acc}]\!] \leftarrow Enc(\epsilon_{acc})$
9 Accumulate noise $\epsilon_{acc} \leftarrow \epsilon_{acc} + \epsilon_m$
10 Send $\frac{\partial \widetilde{L}_n}{\partial w_m} + \epsilon_s$, and $[\![\epsilon_{acc}]\!]$ to the server
11 **Server** S
12 Remove noise $\frac{\partial \widetilde{L}_n}{\partial w_m} \leftarrow \frac{\partial \widetilde{L}_n}{\partial w_m} + \epsilon_s - \epsilon_s$
13 Update $\theta_0 = \{w_1, \cdots, w_m, \alpha_0\}$: $\widetilde{w}_m \leftarrow \widetilde{w}_m - \eta_0 \frac{\partial \widetilde{L}_n}{\partial w_m}$, $\alpha_0 \leftarrow \alpha_0 - \eta_0 \nabla_{\alpha_0} L_n$
14 Remove noise $[\![\frac{\partial L_n}{\partial g_{n,m}}]\!] \leftarrow \frac{\partial \widetilde{L}_n}{\partial g_{n,m}} - [\![\epsilon_{acc}]\!] \cdot \frac{\partial L_n}{\partial z_{n,m}}$
15 Send $[\![\frac{\partial L_n}{\partial g_{n,m}}]\!]$ to client m
16 **Client** $m \in [M]$
17 $\frac{\partial L_n}{\partial g_{n,m}} \leftarrow [Dec([\![\frac{\partial L_n}{\partial g_{n,m}}]\!])$
18 Calculate $\frac{\partial L_n}{\partial \mu_m}$, $\frac{\partial L_n}{\partial \omega_m}$, $\frac{\partial L_n}{\partial \theta_m}$
19 Update $\mu_m \leftarrow \mu_m - \eta_m\big(\frac{\partial L_n}{\partial \mu_m} + \lambda \frac{\partial R_m}{\partial \mu_m}\big)$
20 $\omega_m \leftarrow \omega_m - \eta_m\big(\frac{\partial L_n}{\partial \mu_m} + \lambda \frac{\partial R_m}{\partial \omega_m}\big)$, $\theta_m \leftarrow \theta_m - \eta_m \frac{\partial L_n}{\partial \theta_m}$
21 **Return** the global model $\theta = \{\theta_m\}_{m=0}^M$.

ing a message from a local client m, which is either i) a query of the loss function's gradient w.r.t. to the embedding vector $h_{n,m}$, or ii) a new embedding vector calculated using the updated local model parameter. To respond to query i), the server calculates the gradient for client m using its current $h_{n,m}$, and sends it to client m; and, upon receiving ii), the server computes the new gradient w.r.t. the top model θ_0 using the embedding vectors it currently has from other clients and updates its model. For each interaction with the server, each client m randomly selects a datum $x_{n,m}$, calculates parameters $s_{m,i}$ and $q_{m,j}$, queries the corresponding gradient w.r.t. $h_{n,m}$ from the server, uploads the updated embedding vector $h_{n,m}$, and updates the local bottom model θ_m and variables μ_m and ω_m.

5 Theoretical Analysis

5.1 Convergence Analyses

As illustrated in Section 4.4.2 that FedSDG-FS satisfies the smoothness property and the Assumption 1, and ciphertext operations do not affect the numerical calculations, thus we present the convergence analysis for both FedSDG-FS and FedSDG-FS++ in a unified way. We present the convergence results through two steps. First, we show that there is an equivalence between our proposed l_0 constrained optimization for feature selection and optimization over Bernoulli distribution through Mutual Information (MI). Then, we present the convergence results of the gradient decent methods for optimizing the l_0 constrained optimization. Without loss of generality, we only consider the bottom level stochastic gates. The goal of feature selection is to find the subset of features Q that has the highest MI with the target variable Y. We can formulate the task as selecting Q such that the MI $I(\cdot)$ between X_Q and Y is maximized:

$$\max_{Q} I(X_Q, Y) \quad s.t. \quad |Q| = k. \tag{11}$$

Then, under the mild assumption that there exists an optimal subset of indices Q^*, the Eq.(11) is equivalent to

$$\max_{\mathbf{0}\leq\boldsymbol{\pi}\leq\mathbf{1}} I(X \odot \tilde{Q}; Y) \quad s.t. \quad \sum_i \mathbb{E}\left[\tilde{Q}_i\right] \leq k, \tag{12}$$

where $\tilde{Q}$ are independently sampled from the Bernoulli distribution with parameter $\boldsymbol{\pi}$. Then, we can rewrite this constrained optimization problem as a penalty optimization problem, which is the same as Eq. (5):

$$R = \min L(X \odot \tilde{Q}; Y) + \lambda|\tilde{Q}| \tag{13}$$

So far, we have proved the equivalence between the proposed l_0 constrained optimization and the selection of the optimal feature subset. Then, we proceed to give the convergence results of the l_0 constrained optimization for feature selection. For notational brevity, at iteration t, we define,

$$\mu_0^t := \frac{\partial}{\partial\theta_0} L(h(\theta_0^t, g_{n,1}^t, \cdots, g_{n,M}^t); y_n) \tag{14}$$

$$\mu_m^t := \frac{\partial}{\partial\theta_m} L(h(\theta_0^t, g_{n,1}^t, \cdots, g_{n,M}^t); y_n), \tag{15}$$

Theorem 2. *Under Assumption 1, and the additional assumption that $R(\theta)$ is ρ-strongly convex, if $\eta^t = \frac{1}{\rho \min_m (t+T_0)}$ with the constant $T_0 > 0$. Then the convergence rate is $\mathcal{O}(1/T)$.*

The objective value satisfies the following inequality,

$$\begin{aligned} R(\theta^{t+1}) \leq R(\theta^t) - \eta_0^t(1 - \frac{K}{2}\eta_0^t)||\mu_0^t||^2 \\ - \sum_{m=1}^{M} \eta_m^t(1 - \frac{K_m}{2}\eta_m^t)||\mu_m^t||^2, \end{aligned} \tag{16}$$

where $0 < \eta_0^t \leq \frac{1}{K}, 0 < \eta_m^t < \frac{1}{K_m}$, then $1 - \frac{K}{2}\eta^t \leq \frac{1}{2}, 1 - \frac{K}{2}\eta_m^t \leq \frac{1}{2}$. Then, we have

$$R(\theta^{t+1}) \leq R(\theta^t) - \frac{1}{2}(\eta_0^t||\mu_0^t||^2 + \sum_{m=1}^{M} \eta_m^t||\mu_m^t||^2). \tag{17}$$

We denote $\hat{\theta}$ as the optimal global model. Then, according to the first-order Taylor expansion, we have

$$R(\theta^t) \leq R(\hat{\theta}) + (\mu_0^t + \sum_{m=1}^{M} \mu_m^t)(\theta^t - \hat{\theta}) \tag{18}$$

Substitute the above formula into Eq. (17), we have:

$$R(\theta^t) \leq R(\hat{\theta}) + \frac{1}{2(\eta_0^t + \eta_m^t)T}(||\theta_0 - \hat{\theta}||^2) \tag{19}$$

As T increases, $\epsilon = \frac{1}{2(\eta_0^t+\eta_m^t)T}(||\theta_0 - \hat{\theta}||^2)$ decreases, θ^t gets increasingly close to the optimal solution $\hat{\theta}$, and the convergence rate is $\mathcal{O}(1/T)$.

Thus, we have proved the convergence of the proposed algorithm under the aforementioned assumptions.

Time and storage complexity analysis. The time complexity of the algorithm is $\mathcal{O}(\frac{K||\theta_0-\hat{\theta}||^2}{2\epsilon}), \epsilon = \frac{1}{2(\eta^t+\eta_m^t)T}(||\theta_0 - \hat{\theta}||^2)$. The storage complexity is $\mathcal{O}(|\theta| + |s| + |q|)$, where $|\cdot|$ denotes the parameter size, and the communication cost is $\mathcal{O}\left(|q|\frac{K||\theta_0-\hat{\theta}||^2}{\epsilon}\right)$.

5.2 Privacy Analysis for FedSDG-FS

5.2.1 Privacy Guarantees

We illustrate that FedSDG-FS satisfies β-GDP for the dataset U_m of client m by proving Theory 1. Let $u_m^i, \hat{u}_m^i$ denote the outputs of i-th layer with inputs $u_m^0 = x_{n,m}, \hat{x}_{n,m}$. Under the assumptions that $z_m^i \sim \mathcal{U}[-\zeta_m^i/2, \zeta_m^i/2]$, σ_i is L_{σ_i}-Lipschitz continuous for $i \in [I-1]$, we can derive that,

$$\begin{aligned} ||w_m^I u_m^{I-1} - w_m^I \hat{u}_m^{I-1}|| &\leq ||w_I|| L_{\sigma_{I-1}}(||w_m^{I-1}||\,||u_m^{I-2} - \hat{u}_m^{I-2}||) \\ &\leq ||w_m^I|| \prod_{i=1}^{I-1} L_{\sigma_i} ||w_m^i||\,||x_{n,m} - \hat{x}_{n,m}|| \end{aligned} \tag{20}$$

Consider the linear operation of I-th layer $\mathcal{M}(u_m^{I-1}) = w_m^I u_m^{I-1} + b_m^I + z_m^I$ which is a random mechanism defined by $z_m^I \sim \mathcal{N}(0, \zeta_m^2)$. Since differential privacy is not affected to post-processing [36], $\sigma_I \circ \mathcal{M}$ does not increase the privacy loss with $\mathcal{M}$. According to Theorem 1 in [38], Algorithm 2 is (ε, δ)-differentially private if $\zeta_m = \frac{\sqrt{H\log(1/\delta)}}{\varepsilon}$.

5.2.2 Utility Analyses

We build utility analyses for Algorithm 2 upon the difference bounds of the objective after local perturbation. We evaluate the difference between $R_\zeta(\boldsymbol{\theta}, s, q) = \mathbb{E}_z[R(\boldsymbol{\theta}, s, q)]$ and $R(\boldsymbol{\theta}, s, q)$ (Eq. (5)). Note that,

$$|R_\zeta(\boldsymbol{\theta}, s, q) - R(\boldsymbol{\theta}, s, q)|^2 \leq \frac{M}{N} \sum_{n=1}^{N} L^2 \mathbb{E}_z[||\widetilde{h}_{n,m} - h_{n,m}||]^2 \tag{21}$$

where $h_{n,m}$, $\widetilde{h}_{n,m}$ corresponds to the outputs of Eq. (8), Eq. (9). Since we have that

$$\begin{aligned} ||\widetilde{h}_{n,m} - h_{n,m}|| &= ||\widetilde{u}_I - u_I|| \\ &\leq L_{\sigma_I}(||w_I||\,||\widetilde{u}_I - u_I|| + ||z_I||) \\ &\leq \sum_{i=1}^{I} \Big(\prod_{j=i}^{I} L_{\sigma_j} ||w_m^j||\Big) ||z_j|| \end{aligned} \tag{22}$$

Taking expectation on square of both sides, we have

$$\mathbb{E}||\widetilde{h}_{n,m} - h_{n,m}||^2 \leq \Big(\prod_{j=i}^{I} L_{\sigma_j}^2 ||w_m^j||^2\Big)\Big(\sum_{i=1}^{I-1} \zeta_m^{2i} + \zeta^2\Big) \tag{23}$$

Substituting it into Eq. (21), we arrive that

$$|R_\zeta(\boldsymbol{\theta}, s, q) - R(\boldsymbol{\theta}, s, q)|^2 \leq M\Big(\prod_{j=i}^{I} L_{\sigma_j}^2 ||w_m^j||^2\Big)^{\frac{1}{2}}\Big(\sum_{i=1}^{I-1} \zeta_m^{2i} + \zeta^2\Big)^{\frac{1}{2}} \tag{24}$$

5.3 Security Analysis of FedSDG-FS++

In this section, we perform security analysis of FedSDG-FS++ against the well-known equation solving attack in the semi-honest model [47].

Proposition 1 (Gradient protection in backward propagation)**.** *The server cannot learn the true values of $\frac{\partial L_n}{\partial g_{n,m}}$ in order to reconstruct activations and training data.*

Proof. The prove follows the simulation approach [48], where the basic idea is to build a simulator given the client an input and a global output, and show that it learns nothing but the final result. During model training, the server attempts to remove the randomness from the received gradients. In the space $\mathbb{Z}$, given a noise value ϵ_m selected by client m and a value ϵ_s for the server, the probability that ϵ_m equals ϵ_s is $p\{\epsilon_m = \epsilon_s\} \leq 1 - e^{-2/|\mathbb{Z}|}$, where $|\mathbb{Z}|$ is the size of a finite field identical to the cipher space of Paillier. Since the elements of the noise is independent, the server can correctly yield vectors of $g_{n,m}$ with the probability $p\{g_{n,s} = g_{n,m}\} \leq (1 - e^{-2/|\mathbb{Z}|})^{n_l}$, where n_l is the number of neuron in the last embedding layer. Since $|\mathbb{Z}|$ is a large number, the probability that the server can successfully derive the gradients is close to zero. □

Proposition 2 (Model protection in forward propagation)**.** *The accumulated weighted vectors $\{z_{n,m} = g_{n,m} \cdot w_m\}$, $n \in [N], m \in [M]$ reveals nothing but the sub-spaces of weights from which the weights w_m cannot be reconstructed by clients.*

Proof. Let $z_{n,m} = g_{n,m} \cdot w_m$ denote the weighted vector obtained by client m after one forward propagation. It does not reveal any information regarding the actual values of the matrices w_m but the sub-spaces linearly combined by infinitely many possible matrices solutions. Hence, top model weights w_m cannot be reconstructed by local clients. □

Thus, we can conclude that the server cannot reconstruct training data and clients cannot infer the labels during feature selection and training process.

6 Experimental Evaluation

6.1 Experiment Configuration

1) Datasets. We use 10 datasets with 4 types of data: tabular data, images, texts and audios. These include 2 synthetic datasets, `MADELON` [40] and `FRIEDMAN` [49]; and 8 real-world datasets, `ARCENE` [39], `KDD99` [50], `BASEHOCK` [51], `RELATHE` [51], `PCMAC` [51], `GISETTE` [52], `COIL20` [53] and `ISOLET` [54]. The synthetic datasets are derived from the feature selection challenge [40], where `MADELON` consists of 5 informative features, 15 redundant features constructed by linear combinations of those 5 informative features, and 480 noisy features, while `FRIEDMAN` consists of 5 informative and 995 noisy features. For the real-world datasets, most of them are collected from the ASU feature selection database online [51], and `KDD99` is used for the third international knowledge discovery and data mining tools competition. The descriptions of all datasets are listed in Table 2. We employ two clients in our settings, where we divide features into two parts randomly for every dataset, and assign each part to each client. The labels are located in the server. To further investigate the performance of FedSDG-FS and FedSDG-FS++, we also employ five and ten clients for implementations (see Section 6.3).

TABLE 2
Description of datasets for empirical evaluations

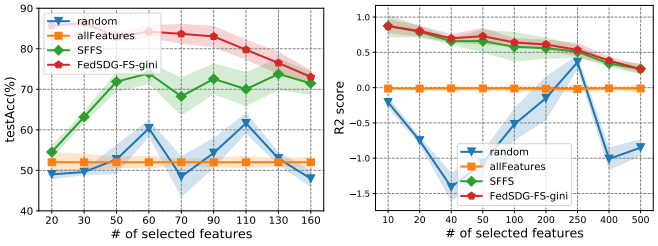
Dataset	Features	Train size	Test size	Classes	Type
MADELON	500	2,000	2,400	2	Tabular
FRIEDMAN	1,000	750	250	2	Tabular
ARCENE	10,000	1,400	600	2	Tabular
KDD99	41	4,898,431	311,029	23	Tabular
BASEHOCK	7,862	1,594	398	2	Text
RELATHE	4,322	2,320	2,088	2	Text
PCMAC	3,289	1,554	388	2	Text
GISETTE	5,000	5,600	1,400	2	Image
COIL20	1,024	1,008	432	20	Image
ISOLET	617	1,248	312	26	Audio

2) VFL Models. We have implemented the typical logistic regression model for VFL [55] on `FRIEDMAN`, and neural networks for VFL [30] on the other 9 datasets (see Table 3). We run VFL models until a pre-specified test accuracy is reached, or a maximum number of iterations has elapsed. In addition, training the dual-gates until convergence may sometimes cause overfitting of the model, where we set the cutoff value of the variables and perform early stopping. We test the accuracy of the global model on the test datasets. We build our VFL models with Flower 0.19.0 [56] and Pytorch 1.8.1 [57]. All the experiments are performed on Ubuntu 16 operating system equipped with a 12-core i7 Intel CPU, 64G of RAM and 4 Titan X GPUs.

3) Comparison Baselines. We compare FedSDG-FS and FedSDG-FS++ methods against state-of-art baselines, **1) allFeatures**: It performs feature selection with all features participating. **2) SFFS** [21]: It performs feature selection

TABLE 3
Settings for training different VFL models.

Model	# of parameters	Task
VFLNN-MADELON	130,552	Two-class classification
VFLLR-FRIEDMAN	130,501	Regression
VFLNN-ARCENE	1,030,552	Cancer detection
VFLNN-KDD99	109,360	Network intrusion detection
VFLNN-BASEHOCK	516,752	Text classification
VFLNN-RELATHE	462,752	Text classification
VFLNN-PCMAC	359,452	Text classification
VFLNN-GISETTE	530,552	Digit number recognition
VFLNN-COIL20	109,360	Face image recognition
VFLNN-ISOLET	93,476	Letter-name recognition



(a) VFLNN-MADELON (b) VFLLR-FRIEDMAN

Fig. 4. Test accuracy and R^2 scores vs. number of selected features on synthetic datasets.

with the filter method based on secure multi-party computation. **3) MS-GINI** [18]: It leverages a filter feature selection method based on Gini impurity. **4) VFLFS** [28]: It leverages the embedded method through combining the auto-encoder with l_2 constraints on feature weights. **5) LESS-VFL** [29]: It utilizes a pre-training step and determines the relevant features by embedding component selection. **6) VertiBench** [35]: It proposes dataset splitting methods based on feature importance and feature correlation to improve model performance. **7) original-Gate** [19]: It has neither gates of the embedding vectors nor importance initialization. For fair comparison, we implement VFLFS [17] without the part that makes use of the non-overlapping samples. We use the Paillier as the PHE method in our implementation.

4) Hyperparameter Setting. We use the Adam optimizer, and set learning rate $\eta = 0.03$, batch size $b = 128$, weight factor $\lambda = 0.1$. We add random noises on the output of each local embedding layer, and the noises follow the uniform distributions $\mathcal{U}[-\sqrt{3}/5, \sqrt{3}/5]$ with $\zeta = 0.2$, $\epsilon = 4$, $\delta = 0.01$. For baseline methods that have their own parameters, we follow the parameter settings in their implementations.

6.2 Evaluating Gini Impurity for VFL

Firstly, we evaluate the effectiveness of our Gini impurity metric (FedSDG-FS-gini) designed for feature importance initialization in both the FedSDG-FS and FedSDG-FS++ by comparing the test accuracy of the global models to the other three filtering based feature selection strategies, SFFS [21], random FS, and all features participating (allFeatures). We select different numbers of features (*i.e.*, k features with the smallest Gini scores), and assign them to the two clients. Since those datasets differ in both the number of features and the number of noisy features. Thus, we select features from each dataset in similar proportions and round the numbers of selected features. We perform 5-fold cross validation and report the average R^2 scores, test accuracy and standard deviations in Fig. 4. Here, the R^2 score is defined as $R^2 = 1 - \frac{\sum_{n\in[N]}(y_n-\hat{y}_n)^2}{\sum_{n\in[N]}(y_n-\bar{y})}$, where $\bar{y} = \frac{1}{N}\sum_{n\in[N]} y_n$, and y_n, $\hat{y}_n$ are the true target and predicted target of the n-th sample, respectively. The results show that FedSDG-FS-gini achieves higher test accuracy and R^2 scores than other strategies. Specifically, the average test accuracy and R^2 scores of FedSDG-FS-gini are 28.71%/ 123.8%, 29.69%/ 70.3%, 12.85%/ 3.4% higher than that of random, allFeatures and SFFS for MADELON and FRIEDMAN, respectively. Meanwhile, the standard deviations are relative small, *e.g.*, with 1.22% and 4.3% smaller than that of SFFS for MADELON and FRIEDMAN. In addition, we analyze why some of the test accuracy in Fig 4 is less than 50%. First, since there are noisy features irrelevant to the learning task and a large number of redundant features possessed by local clients, samples with very similar or the same features may have completely opposite labels. Such low quality training data makes the models perform almost like random guesses or slightly worse when they overfit the noise.

We further evaluate the performance of FedSDG-FS when only a subset of clients participates in the feature importance initialization. Specifically, we train different VFL models (illustrated in Table 3) on corresponding datasets of clients with different selection ratios, *i.e.*, $r = 0.1$, 0.3, 0.5, 0.7, 1.0. For the remaining clients who did not participate in the feature importance initialization, we assign their feature importance scores to be zeros. The test accuracy results are shown in Table 5 and Fig. 5. It can be observed that as the selection ratio decreases, the test accuracy decreases slightly, and the model converges slower. For example, for the BASEHOCK and RELATHE dataset, when the participating ratio is 0.5, the drop of test accuracy is much small, only 0.06%, 0.57%. To solve this problem, we need to allow all clients to participate in the initialization stage, which is much efficient and incremental, *i.e.*, clients can individually interact with the server to perform the feature importance initialization without recomputing feature importance from scratch. Take the dataset ARECENE as an example, the total communication overhead is 7.82 MB and the computation cost is 173.36s, which is much efficient to calculate.

6.3 Evaluating Important Feature Selection

After feature importance initialization, the server proceeds to select important features using the stochastic dual-gates.

1) Learning Accuracy. We compare FedSDG-FS and FedSDG-FS++ with other baselines by training different VFL models and evaluating the test accuracy of the global models, and the ratios of selected features. The results are shown in Table 4. It can be observed that FedSDG-FS++ achieves the highest test accuracy using the fewest features in almost all datasets, and FedSDG-FS achieves the comparable test accuracy with similar number of features. Taking MADELON as an example, the average test accuracy of FedSDG-FS++ is 47.2%, 33.6%, 27.0%, 48.2%, 13.39%, 7.7%, 0.3/%, 0.1%, higher than other seven baselines, respectively; while the ratio of selected features is 0.97, 0.47, 0.47, 0.47, 0.54, 0.78, 0.02, 0.01, less than them. Fig. 6 shows the ROC curve of FedSDG-FS++, and the ROC curve of FedSDG-FS and on other datasets is much similar to Fig. 6. It also illustrates that our methods achieve high test accuracy on various datasets. In some cases where there are small number of noisy

TABLE 4
Test accuracy of models trained with features selected with different methods.

Datasets	Test Accuracy (%) / Ratio of Selected Features								
	allFeatures	SFFS	MS-GINI	VFLFS	LESS-VFL	VertiBench	original-Gate	FedSDG-FS	FedSDG-FS++
MADELON	52.0/ 1.0	65.6/ 0.5	72.2/ 0.5	51.0/ 0.5	85.81/ 0.57	91.5/ 0.81	98.9/ 0.05	99.1/0.04	**99.2**/ 0.03
ARCENE	80.1/ 1.0	95.0/ 0.5	87.5/ 0.5	70.1/ 0.5	90.1/ 0.53	92.31/ 0.67	97.4/ 1.0	99.0/0.57	**99.8**/ 0.58
KDD99	68.7/ 1.0	80.1/ 0.5	69.5/ 0.5	67.1/ 0.5	72.31/ 0.66	69.21/ 0.72	75.4/ 1.0	77.9/0.59	**78.8**/ 0.54
BASEHOCK	99.7/ 1.0	99.1/ 0.5	98.5/ 0.5	94.4/ 0.5	98.71/ 0.54	98.89/ 0.58	99.5/ 0.48	99.7/0.35	**99.9**/ 0.3
RELATHE	95.5/ 1.0	87.2/ 0.5	92.1/ 0.5	86.5/ 0.5	94.78/ 0.71	93.78/ 0.58	99.7/ 0.71	99.6/0.41	**99.8**/ 0.41
PCMAC	97.6/ 1.0	79.34/ 0.5	90.2/ 0.5	86.11/ 0.5	92.67/ 0.54	93.15/ 0.62	**99.1**/ 0.66	98.7/0.55	98.7/ 0.45
GISETTE	99.1/ 1.0	99.0/ 0.5	98.0/ 0.5	50.2/ 0.5	99.3/ 0.81	96.8/ 0.68	97.8/ 0.76	99.5/0.57	**99.5**/ 0.53
COIL20	96.4/ 1.0	65.6/ 0.5	94.8/ 0.5	72.2/ 0.5	89.14/ 0.62	87.76/ 0.73	91.2/ 1.0	97.3/0.70	**97.5**/ 0.71
ISOLET	**98.0**/ 1.0	92.7/ 0.5	91.4/ 0.5	71.4/ 0.5	93.21/ 0.63	91.05/ 0.57	93.2/ 1.0	94.8/ 0.76	96.7/ 0.75

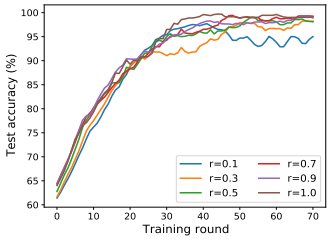


Fig. 5. Test accuracy of VFLNN−ARCENE with different ratio of clients.

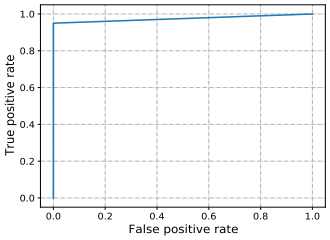


Fig. 6. ROC curve of the model VFLNN−ARCENE.

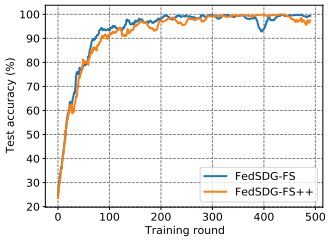


Fig. 7. Test accuracy of model VFLNN−COIL.

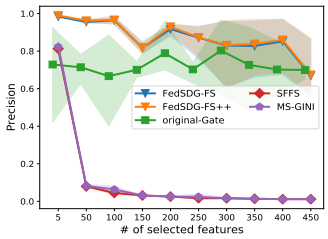
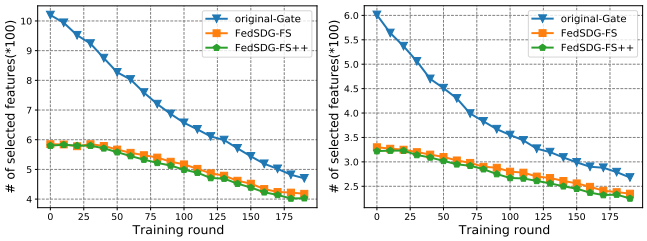
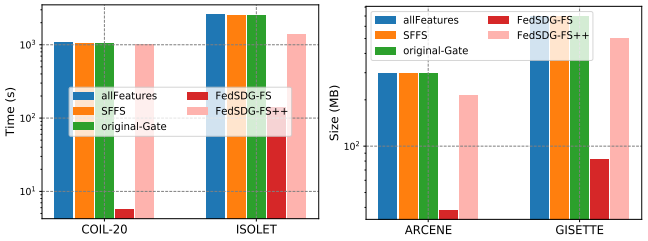


Fig. 8. Precision of different methods on MADELON.



(a) VFLNN−COIL (b) VFLNN−ISOLET

Fig. 9. The number of features selected of different training rounds of VFLNN−COIL and VFLNN−ISOLET.



(a) Computation cost (s) (b) Communication cost (MB)

Fig. 10. Computation cost and communication cost of model inference for the samples of test datasets ARCENE and GISETTE.

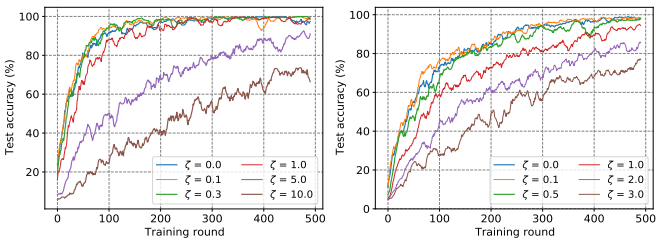
TABLE 5
Test accuracy of models trained with FedSDG-FS with different ratios (%) of clients participating in feature importance initialization.

Datasets	Test accuracy (%)/ ratio of selected features					
	1.0	0.9	0.7	0.5	0.3	0.1
ARCENE	99.80	99.68	98.80	97.64	97.23	96.89
BASEHOCK	99.65	99.61	99.63	99.59	99.42	99.10
RELATHE	99.04	98.89	98.28	98.47	98.32	98.11
PCMAC	98.70	98.25	98.13	98.05	97.85	97.67

features, and having little negative impact on the model, using all features results the higher accuracy. Nevertheless, FedSDG-FS and FedSDG-FS++ can still achieve comparable test accuracy with fewer features. In addition, FedSDG-FS++ is a lossless solution since it adopts the strategy that sending activation back and forth between the client and the server for handling non-linear operation instead of using the polynomial function approximation method [58]. Besides, we show the results of test accuracy different communication rounds for training VFLNN−COIL model using the FedSDG-FS and FedSDG-FS++ in Fig. 7. It can be observed that FedSDG-FS and FedSDG-FS++ achieve much similar test accuracy during training process with small DP noises, *i.e.*, $(10^{-4}, 0.01)$.

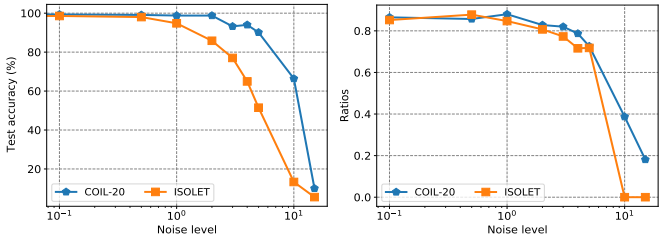
2) Precision. We use precision to measure the accuracy of FedSDG-FS and FedSDG-FS++, which calculates the proportion of correctly selected informative features over all selected features. For FedSDG-FS and the original gate method, we train VFLNN−MADELONE until the model converges, and determine the important features. For SFFS and MS-GINI, we calculate the F-statistics and Gini impurity of each individual feature, respectively, and select different numbers of informative features. The results are shown in Fig 8. It can be observed that FedSDG-FS, FedSDG-FS++ and the original gate method achieve much higher precision than allFeatures, SFFS, MS-GINI, and FedSDG-FS++ achieves the highest precision. This illustrates that reducing the sizes of embedding vectors does not degrade the model accuracy. For example, the average precision scores of different number settings of FedSDG-FS++ are 13% and 76% higher than the original gate method and SFFS, respectively. As the number of selected features increases, the precision of SFFS and MS-GINI decreases dramatically, while the presicion of FedSDG-FS, FedSDG-FS++ and original gate methods decreases slightly, which demonstrates their effectiveness without knowing the number of features to be selected.

3) Efficiency. We evaluate the efficiency of FedSDG-FS and FedSDG-FS++ from two aspects: 1) the speed of



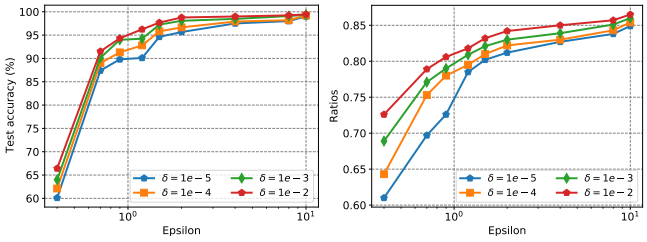
(a) VFLNN-COIL (b) VFLNN-ISOLET

Fig. 11. Test accuracy under different noise levels, *i.e.*, different ζ values of FedSDG-FS.



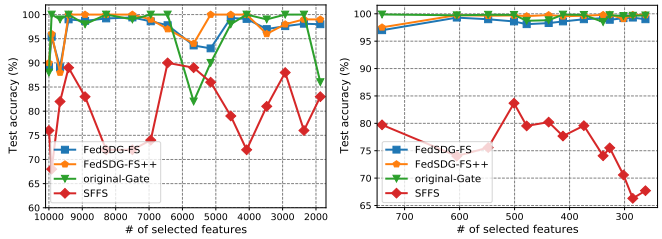
(a) Test accuracy (b) Ratio of same selected features

Fig. 12. Test accuracy and ratios of same selected features under different noise levels, *i.e.*, different ζ values of FedSDG-FS.



(a) Test accuracy (b) Ratio of same selected features

Fig. 13. Test accuracy and ratios of same selected features of FedSDG-FS with various (ϵ, δ) privacy values on the dataset COIL-20.



(a) VFLNN-ARCENE (b) VFLNN-GISETTE

Fig. 14. Test accuracy vs. number of selected features by training models for 20 rounds.

TABLE 6
Test accuracy of models trained with features selected with different methods.

# clients	Dataset	Test accuracy (%) / Ratio of Selected Features						
		allFeatures	SFFS	MS-GINI	VFLFS	original-Gate	FedSDG-FS	FedSDG-FS++
5	ARCENE	85.8/ 1.0	92.2/ 0.5	92.0/ 0.5	71.0/ 0.5	98.2/ 1.0	98.7/ 0.59	**99.7**/ 0.59
	RELATHE	97.6/ 1.0	84.8/ 0.5	92.7/ 0.5	86.1/ 0.5	**99.8**/ 0.69	99.3/ 0.38	99.5/ 0.45
10	ARCENE	91.0/ 1.0	94.0/ 0.5	92.7/ 0.5	82.0/ 0.5	98.6/ 1.0	99.0/ 0.57	**99.2**/ 0.56
	RELATHE	97.5/ 1.0	85.6/ 0.5	93.6/ 0.5	85.7/ 0.5	99.5/ 0.68	98.9/ 0.40	**99.8**/ 0.44
100	ARCENE	80.7/ 1.0	89.1/ 0.5	89.4/ 0.5	70.2/ 0.5	97.8/ 0.61	98.2/ 0.61	**98.8/ 0.57**
	RELATHE	95.5/ 1.0	82.6/ 0.5	91.4/ 0.5	81.2/ 0.5	98.5/ 0.58	97.8/ 0.40	**98.6**/ 0.44

the feature importance initialization, and 2) computation and communication saving during model prediction. Firstly, we calculate the number of selected features in different training rounds. Take training models VFLNN-COIL and VFLNN-ISOLET (Fig. 9(a) and Fig. 9(b)) for example. Here, we set the weight factor $\lambda = 1.0$. The results show that with importance initialization, FedSDG-FS, FedSDG-FS++ can filter out noisy features before training, and can quickly filter out noisy features and select important ones, thus speeding up model training. Secondly, we compare the prediction computation overhead and communication overhead of those models of FedSDG-FS, FedSDG-FS++ and other baseline methods. For example, Fig. 10 shows the total computation and communication cost of each method to perform model inference on test datasets of ARCENE and GISETTE. The computation cost of FedSDG-FS is much lower than other methods, *e.g.*, 99.46%, 99.45%, 99.44%, 99.42% lower of dataset ARCENE, and the computation cost of FedSDG-FS++ is 6.69%, 4.18%, 2.61% lower than allFeatures, SFFS and original-Gate methods. Besides, FedSDG-FS saves 87.11%, 86.12%, 87.11%, 82.06% communication cost, while FedSDG-FS++ saves 28.13%, 28.21%, 28.56% communication cost. The efficiency analysis clearly demonstrated the advantages of the feature importance initialization module and the DP-based feature selection module.

6.4 Differential Privacy Analysis

To illustrate that our differentially private FedSDG-FS is the small difference in model testing accuracy as well as small differences in selected features compared to the non-private system, we first train various models under different noise levels, *i.e.*, with different ζ values and report the test accuracy in different training rounds. We refer to FedSDG-FS++ as the setting with $\zeta = 0$. The example results of models VFLNN-COIL and VFLNN-ISOLET are shown in Fig. 11 and Fig. 12. It can be observed that with some moderate noises, the difference in test accuracy is much small, *e.g.*, 0.45%, 0.2% with $\zeta = 0.1$, and 0.7%, 0.76% with $\zeta = 0.5$ for datasets COIL-20 and ISOLET. Besides, the results show that the ratio of the same selected features are much high, *e.g.*, 0.865, 0.852 with $\zeta = 0.1$, and 0.857, 0.878 with $\zeta = 0.5$ for datasets COIL-20 and ISOLET. In addition, the noise levels affect the performance of feature selection and model training. Specifically, as noise increases, the ratio of the same selected features and model test accuracy decrease dramatically, and the model converges much slower. Take dataset COIL-20 for example, when the noise value increases from 1.0 to 15.0, the test accuracy drops from 98.76% to 10.1%, and the ratio of the same features drops from 0.82 to 0.18 accordingly. For dataset ISOLTE, when the noise equals 10 or 15, the model learns almost

nothing, thus the ratio of the same selected features is 0. By using the moments accountant, we can obtain a δ value for any given ϵ. We record the test accuracy for different (ϵ, δ) pairs in Fig. 13. In the figure, each curve corresponds to the average test accuracy of models achieved for a fixed δ, as it varies between 10^{-5} and 10^{-2}. For example, we can achieve 91.52% accuracy for VFLNN-COIL with $\epsilon = 0.7$ and $\delta = 0.01$. In addition, with strong privacy guarantees, *i.e.*, with ϵ being 0.4, the test accuracy decreases dramatically. Thus, by adjusting the added noise, we can balance the feature selection performance and privacy protection level.

6.5 Ablation Study

1) Impacts of number of selected features. We evaluate the stability of FedSDG-FS by calculating the test accuracy of the global model with different numbers of selected features. We illustrate the test accuracy of models VFLNN-ARCENE and VFLNN-GISETTE by training them for 20 rounds with different numbers of features in Fig. 14. The results show that compared to SFFS, the original based method, FedSDG-FS and FedSDG-FS++ all achieve much higher test accuracy. Meanwhile, the performance of FedSDG-FS and FedSDG-FS++ has less variation in all cases than other two methods.

2) Impacts of number of participants. To further validate the proposed methods, we employ larger number of participants, *i.e.*, 5 clients, 10 clients and 100 clients for model training. Two example results on datasets ARCENE and RELATHE are presented in Table 6. It shows that as the number of clients increases, the test accuracy of the global model decreases slightly, which demonstrates the scalability of our methods. Besides, FedSDG-FS and FedSDG-FS++ achieve comparable high test accuracy, and FedSDG-FS++ achieves the highest test accuracy using the fewest features in most cases. For the only case where FedSDG-FS++ performs second best in terms of accuracy, our accuracy 99.5% is very close to the best accuracy 99.8%, while FedSDG-FS++ use about 20% fewer features.

3) Impacts of initialization metrics. One of our contributions is designing a feature importance initialization method based on Gini impurity and making it work in a privacy-preserving manner under VFL settings. It is not easy to design a feature selection approach which is also privacy preserving. We compare our feature importance initialization metric Gini with other metrics, including F-statistics and Mutual Information (MI) without considering the privacy issue. Table 7 shows that these methods achieve comparable feature selection performance. We will explore how to make the information gain based feature selection method work in VFL settings in our future work.

TABLE 7
Test accuracy of models trained with features selected with different importance initialization methods.

Datasets	Test accuracy (%)/ ratio of selected features			
	F-statistics	MI	Lasso	Gini
ARCENE	98.4/ 0.59	99.8/ 0.61	99.1/ 0.59	99.8/ 0.58
KDD99	78.2/ 0.51	77.93/ 0.48	79.0/ 0.42	78.8/ 0.54
RELATHE	99.8/ 0.41	99.3/ 0.46	99.5/ 0.39	99.8/ 0.41
COIL20	97.8/ 0.42	97.92/ 0.43	96.53/ 0.51	98.7/ 0.45
ISOLET	95.9/ 0.69	96.3/ 0.72	97.1/ 0.68	96.7/ 0.71

7 Conclusions and Future Work

In this work, we proposed an efficient and privacy-preserving vertical federated learning feature selection framework to select important features in VFL settings. We first designed a Gaussian stochastic dual-gates for clients' inputs to efficiently approximate the probability of a feature being selected. To adaptively fulfill the two-level privacy requirements, we first proposed a local embedding perturbation approach, FedSDG-FS, to achieve differential privacy for local training data. Then, we proposed FedSDG-FS++ which incorporates PHE and randomized noise mechanism into the stochastic dual-gates to achieve secure feature selection. To reduce overhead, we proposed a feature importance initialization method based on Gini impurity and PHE, which can be accomplished through only two parameter transmissions, and two encryption/decryption operation on the server. Experiment results show that the proposed methods significantly outperform existing approaches in terms of achieving more accurate selection of high-quality features and building global models with better performance.

Acknowledgments

Han Yu is the corresponding author of this work. This research is supported by Nanyang Technological University (NTU), under SUG Grant (020724-00001); the National Research Foundation, Prime Ministers Office, National Cybersecurity R&D Program (No. NRF2018NCR-NCR005-0001), NRF Investigatorship NRF-NRFI06-2020-0001; the National Research Foundation, Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-019); Alibaba Group through Alibaba Innovative Research (AIR) Program and Alibaba-NTU Singapore Joint Research Institute (JRI) (Alibaba-NTU-AIR2019B1), NTU, Singapore; the RIE 2020 Advanced Manufacturing and Engineering Programmatic Fund (No. A20G8b0102), Singapore; the National Key R&D Program of China 2021YFB2900103, China National Natural Science Foundation with No. 61932016, and "the Fundamental Research Funds for the Central Universities" WK2150110024.

References

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

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

[3] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, *Federated Learning*. Springer, Cham, 2020, vol. Synthesis Lectures on Artificial Intelligence and Machine Learning.

[4] A. Li, L. Zhang, J. Wang, F. Han, and X.-Y. Li, "Privacy-preserving efficient federated-learning model debugging," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 10, pp. 2291–2303, 2021.

[5] J. Wang, L. Zhang, A. Li, X. You, and H. Cheng, "Efficient participant contribution evaluation for horizontal and vertical federated learning," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 911–923.

[6] R. Goebel, H. Yu, B. Faltings, L. Fan, and Z. Xiong, *Trustworthy Federated Learning*. Springer, Cham, 2023, vol. 13448.

[7] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang, "Fedvision: An online visual object detection platform powered by federated learning," in *IAAI*, 2020, pp. 13 172–13 179.

[8] Z. Liu, Y. Chen, Y. Zhao, H. Yu, Y. Liu, R. Bao, J. Jiang, Z. Nie, Q. Xu, and Q. Yang, "Contribution-aware federated learning for smart healthcare," in *IAAI*, 2022, pp. 12 396–12 404.

[9] Y. Chen, Z. Chen, S. Guo, Y. Zhao, Z. Liu, P. Wu, C. Yang, Z. Li, and H. Yu, "Efficient training of large-scale industrial fault diagnostic models through federated opportunistic block dropout,," in *IAAI*, 2023.

[10] A. Li, L. Zhang, J. Wang, J. Tan, F. Han, Y. Qin, N. M. Freris, and X.-Y. Li, "Efficient federated-learning model debugging," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 372–383.

[11] W. Zhuang, Y. Wen, and S. Zhang, "Joint optimization in edge-cloud continuum for federated unsupervised person re-identification," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 433–441.

[12] A. Li, L. Zhang, J. Tan, Y. Qin, J. Wang, and X.-Y. Li, "Sample-level data selection for federated learning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[13] T. Chen, X. Jin, Y. Sun, and W. Yin, "Vafl: a method of vertical asynchronous federated learning," *arXiv preprint arXiv:2007.06081*, 2020.

[14] J. Tan, L. Zhang, Y. Liu, A. Li, and Y. Wu, "Residue-based label protection mechanisms in vertical logistic regression," *arXiv preprint arXiv:2205.04166*, 2022.

[15] PowerFL, "Angel powerfl," https://data.qq.com/powerfl/.

[16] FATE, "Fate-federated-ai," https://github.com/FederatedAI/DOC-CHN.

[17] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," *arXiv preprint arXiv:1712.01312*, 2017.

[18] X. Li, R. Dowsley, and M. De Cock, "Privacy-preserving feature selection with secure multiparty computation," *ICML 2021*, 2021.

[19] Y. Yamada, O. Lindenbaum, S. Negahban, and Y. Kluger, "Feature selection using stochastic gates," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 648–10 659.

[20] J. Chen, M. Stern, M. J. Wainwright, and M. I. Jordan, "Kernel feature selection via conditional covariance minimization," *NeurIPS 2017*, 2017.

[21] F. Pan, D. Meng, Y. Zhang, H. Li, and X. Li, "Secure federated feature selection for cross-feature federated learning," 2020.

[22] L. Song, A. Smola, A. Gretton, J. Bedo, and K. Borgwardt, "Feature selection via dependence maximization." *Journal of Machine Learning Research*, vol. 13, no. 5, 2012.

[23] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Transactions on neural networks*, vol. 20, no. 2, pp. 189–201, 2009.

[24] D. Roy, K. S. R. Murty, and C. K. Mohan, "Feature selection using deep neural networks," in *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–6.

[25] M. M. Kabir, M. M. Islam, and K. Murase, "A new wrapper feature selection approach using neural network," *Neurocomputing*, vol. 73, no. 16-18, pp. 3273–3283, 2010.

[26] Y. Li, C.-Y. Chen, and W. W. Wasserman, "Deep feature selection: theory and application to identify enhancers and promoters," *Journal of Computational Biology*, vol. 23, no. 5, pp. 322–336, 2016.

[27] C. Hans, "Bayesian lasso regression," *Biometrika*, vol. 96, no. 4, pp. 835–845, 2009.

[28] S. Feng, "Vertical federated learning-based feature selection with non-overlapping sample utilization," *Expert Systems with Applications*, p. 118097, 2022.

[29] T. Castiglia, Y. Zhou, S. Wang, S. Kadhe, N. Baracaldo, and S. Patterson, "Less-vfl: Communication-efficient feature selection for vertical federated learning," *arXiv preprint arXiv:2305.02219*, 2023.

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

[31] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, D. Papadopoulos, and Q. Yang, "Secureboost: A lossless federated learning framework," *IEEE Intelligent Systems*, vol. 36, no. 6, pp. 87–98, 2021.

[32] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[33] M. A. Hall and L. A. Smith, "Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper." in *FLAIRS conference*, vol. 1999, 1999, pp. 235–239.

[34] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini, "Group sparse regularization for deep neural networks," *Neurocomputing*, vol. 241, pp. 81–89, 2017.

[35] Z. Wu, J. Hou, and B. He, "Vertibench: Advancing feature distribution diversity in vertical federated learning benchmarks," *arXiv preprint arXiv:2307.02040*, 2023.

[36] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[37] C. Dwork and Kenthapadi, "Our data, ourselves: Privacy via distributed noise generation," in *EUROCRYPT 2006*, ser. Lecture Notes in Computer Science, May 2006.

[38] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[39] J. Thomas, "Mass spectrometric data." [Online]. Available: https://www.openml.org/d/41157

[40] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the nips 2003 feature selection challenge," *Advances in neural information processing systems*, vol. 17, 2004.

[41] O. Li, J. Sun, X. Yang, W. Gao, H. Zhang, J. Xie, V. Smith, and C. Wang, "Label leakage and protection in two-party split learning," *arXiv preprint arXiv:2102.08504*, 2021.

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

[43] A. Miller, N. Foti, A. D'Amour, and R. P. Adams, "Reducing reparameterization gradient variance," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[44] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *International symposium on privacy enhancing technologies symposium*. Springer, 2009, pp. 235–253.

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

[46] J. Dong, A. Roth, and W. J. Su, "Gaussian differential privacy," *arXiv preprint arXiv:1905.02383*, 2019.

[47] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis." in *USENIX security symposium*, vol. 16, 2016, pp. 601–618.

[48] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[49] J. H. Friedman, "Multivariate adaptive regression splines," *The annals of statistics*, vol. 19, no. 1, pp. 1–67, 1991.

[50] T. U. K. Archive, "Kdd cup 1999 data data set," https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[51] A. state university, "Feature selection datasets," Public online, 2010. [Online]. Available: https://jundongl.github.io/scikit-feature/OLD/datasets_old.html

[52] U. machine learning repository, "Handwritten digit recognition problem." [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Gisette

[53] C. University, "Image classification task." [Online]. Available: https://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[54] U. machine learning repository, "Letter-name classification task." [Online]. Available: https://archive.ics.uci.edu/ml/datasets/isolet

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

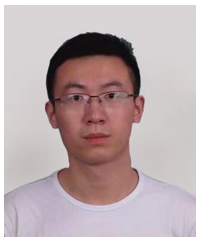
[56] Flower, "Flower: A friendly federated learning framework," Public online, 2022. [Online]. Available: https://flower.dev/

[57] Pytorch, "Pytorch," Public online, 2022. [Online]. Available: https://pytorch.org/

[58] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *International conference on machine learning*. PMLR, 2016, pp. 201–210.

Anran Li is a research fellow in the School of Computer Science and Engineering, Nanyang Technological University, Singapore. She received her Ph.D. degree from University of Science and Technology of China, China, in 2021. Her research interests mainly focus on data quality assessment, federated learning and mobile computing.

Jiahui Huang is a master student of the School of Computer Science and Technology, University of Science and Technology of China, China. He received his Bachelor of Engineering degree from University of Science and Technology of China in 2022. His research interests include federated learning and data quality assessment.



Ju Jia is currently an Associate Professor at the School of Cyber Science and Engineering, Southeast University, China. He received his Ph.D. degree in cyberspace security from Wuhan University, China, in 2021, and he was a research fellow with the School of Computer Science and Engineering at Nanyang Technological University, Singapore. His research interests include: data security, multimedia data security, and artificial intelligence security.

Hongyi Peng is a Ph.D. candidate of Alibaba Talent Programme in the School of Computer Science and Engineering, Nayang Technological University, Singapore. He received his Bachelor of Engineering degree from Nanyang Technological University in 2019 and Master of Science degree from the National University of Singapore in 2020. His research interests include federated learning and data analysis.

Lan Zhang is currently a Professor at the School of Computer Science and Technology, University of Science and Technology of China. She received her Ph.D degree and Bachelor degree from Tsinghua University, China. Her research interests include mobile computing, privacy protection, and data sharing and trading.

Luu Anh Tuan is currently an Assistant Professor at Nanyang Technological University, Singapore. Prior than that, he was a Research Fellow at MIT from 2018 to 2020. Luu's research interests lie in the intersection of AI, Deep Learning, and NLP. Dr. Luu also served as the Senior Area Chair of EMNLP 2020, Area Chair of ACL 2021-2023, Area Chair of ICLR 2022, Area Chair of NeurIPS 2023, Senior Program Committee of IJCAI 2020-2021, and Program Committee member of ICLR, ICML, AAAI, etc.

Han Yu is a Nanyang Assistant Professor (NAP) in the School of Computer Science and Engineering (SCSE), Nanyang Technological University (NTU), Singapore. He held the prestigious Lee Kuan Yew Post-Doctoral Fellowship (LKY PDF) from 2015 to 2018. He obtained his PhD from the School of Computer Science and Engineering, NTU. His research focuses on federated learning and algorithmic fairness. He has published over 200 research papers and book chapters in leading international conferences and journals. He is a co-author of the book *Federated Learning* - the first monograph on the topic of federated learning. His research works have won multiple awards from conferences and journals. He is a *Distinguished Member* of CCF, and a *Senior Member* of AAAI and IEEE.

Xiang-Yang Li is a Full Professor and the Executive Dean of the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. He is an IEEE/ACM Fellow. He received the bachelor degree from the Department of Computer Science, the bachelor degree from the Department of Business Management, Tsinghua University, in 1995, and the Ph.D. degree from the University of Illinois at Urbana–Champaign. His research interests include wireless networking/mobile computing/RFID, privacy and security, cyber-physical systems and IoT, social computing, and interdisciplinary research.