

FedSDG-FS: Efficient and Secure Feature Selection for Vertical Federated Learning

Anran Li, Hongyi Peng, Lan Zhang, Jiahui Huang, Qing Guo, Han Yu, Yang Liu

Abstract—Vertical Federated Learning (VFL) enables multiple data owners, each holding a different subset of features about largely overlapping sets of data sample(s), to jointly train a useful global model. Feature selection (FS) is important to VFL. It is still an open research problem as existing FS works designed for VFL either assumes 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, with privacy protection through Partially Homomorphic Encryption without a trusted third-party. 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. Extensive experiments on both synthetic and real-world datasets show that FedSDG-FS significantly outperforms existing approaches in terms of achieving accurate selection of high-quality features as well as building global models with improved performance.

Index Terms—Feature selection, vertical federated learning

I. INTRODUCTION

Federated learning (FL) [1]–[5] is an emerging machine learning paradigm, which enables multiple data owners to jointly train a model by iteratively exchanging model parameters through an FL server, while preserving local data privacy. Based on the distribution of local data, there are two main categories of FL scenarios: 1) horizontal federated learning (HFL) and 2) vertical federated learning (VFL). Under HFL [6]–[8], data owners’ local datasets have little overlap in the sample space but large overlaps in the feature space. Under VFL [9]–[11], data owners’ local datasets have large overlaps in the sample space but little overlap in the feature space. VFL scenarios often arise in real applications [12], [13], e.g., an e-commerce company, a bank and a ride-sharing company can collaborate to build a model to identify potential financial fraudsters based on the multiple perspectives on people’s behaviour through VFL. The quality of data owners’ local features determines the effectiveness of their local models, thereby affecting the performance of the global VFL model. In practice, data owners can possess noisy features that are irrelevant to the learning task, or a large number of redundant features, which seriously impairs global model performance.

[§]This paper has been accepted by IEEE INFOCOM 2023.

Anran Li, Hongyi Peng, Han Yu and Yang Liu are with the school of Computer Science and Engineering of Nanyang Technological University, Singapore. Lan Zhang and Jiahui Huang are with the school of Computer Science and Technology of University of Science and Technology of China, China. Qing Guo is with the Center for Frontier AI Research, A*STAR, Singapore.

As an example, one of our experiments in Section III-B shows that a two-class classifier trained by VFL with the real dataset suffered an accuracy loss from 82.6% to 54.2% due to the existence of noisy features.

To improve the performance of VFL systems, in this work, we focus on filtering noisy features and selecting important features. A number of feature selection methods have been proposed for centralized machine learning settings [14]–[16], while few work focused on VFL [17]. Feature selection methods for centralized machine learning can be divided into three categories: 1) **filter methods** calculate per-feature relevance scores based on statistical measures (e.g., Gini impurity) to filter features prior to learning a model [16], [18], [19]; 2) **wrapper methods** search for the optimal feature subset in large search spaces [20], [21]; and 3) **embedded methods** attempt to select subset of important features while simultaneously learn the model [15], [22], [23].

Existing FS works designed for VFL either assumes prior knowledge on the number of noisy features [17] or prior knowledge on the post-training threshold of useful features to be selected [24]. 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 in a privacy-preserving manner?* Existing FS methods require direct access to training samples, the training process and the labels simultaneously, which is not permitted in VFL. Besides, during VFL training, intermediate parameters are transmitted in ciphertexts [25], [26], which further increases the difficulty of feature selection. 2) *How to conduct feature selection efficiently and adaptively in VFL settings?* Existing FS methods require a large number of training iterations to select features, especially for high-dimensional data [15], [22]. Directly applying them in VFL will incur significant computation and communication overhead since each training round involves multiple encryption/decryption operations and intermediate parameter transfers.

To address the aforementioned questions and the limitations of existing works [17], [24], we propose the Federated Stochastic Dual-Gate based Feature Selection (FedSDG-FS) approach. It is an embedded feature selection approach consisting of a feature importance initialization module and a secure important feature selection module. Its advantages are summarized as follows:

- **Context-Awareness:** FedSDG-FS can jointly perform feature selection and model training following the proposed stochastic dual-gate and Gini impurity-based feature importance initialization, thereby ensuring the selected features be relevant to the context of the model.
- **Efficiency:** The FedSDG-FS Gini impurity based feature importance initialization enables the global model to quickly filter out noisy features and select important ones, thus speeding up model training. The stochastic dual-gates are designed to reduce the sizes of the embedding vectors, thereby saving communication costs.
- **Security:** FedSDG-FS achieves secure feature selection and model training by leveraging partially homomorphic encryption (PHE) and the randomized mechanism. During the feature selection and model training process, neither data nor labels are exposed to any party other than their original owners.

We evaluate FedSDG-FS via extensive experiments using nine datasets including tabular data, images, texts and audios on a VFL system. The results show that it significantly outperforms existing approaches in terms of achieving accurate and secure 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.

II. RELATED WORKS

Feature selection plays an important role in machine learning tasks. There are a number of feature selection methods proposed for centralized machine learning settings [14]–[16], while few works deal with feature selection in VFL [17].

A. 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. Filter FS methods attempt to remove irrelevant features prior to learning a model. These methods filter features using relevance scores (*e.g.*, Gini impurity) and mutual information, which are calculated based on statistical measures [14], [16], [18], [27]. Wrapper FS 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 [20], [21], [28]. Embedded FS methods aim to select a subset of relevant features, while simultaneously learning the model [15], [22], [23]. The least absolute shrinkage and selection operator [23] is a well-known embedded FS method, whose objective is to minimize the loss while enforcing an l_1 constraint on the weights of the features. Another recently proposed method [15] uses a continuously relaxed Bernoulli variable to conduct FS based on stochastic gates. However, this method requires a large number of parameters to be trained

in the first layer, 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. In addition, they are also not optimized to reduce communication or computation costs when the volume of training data is large.

B. Feature Selection in VFL

In VFL, there are only two works on feature selection (FS) [17], [29]. In [17], FS is performed 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 FS. 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. In [29] the embedded method combined the auto-encoder with l_2 constraints on feature weights is used for FS. However, it suffers from shrinkage of the model parameters, and requires post-training threshold setting to determine the selected features [24]. The proposed FedSDG-FS approach addresses these limitations of the state of the art.

III. PRELIMINARIES & PROBLEM DEFINITION

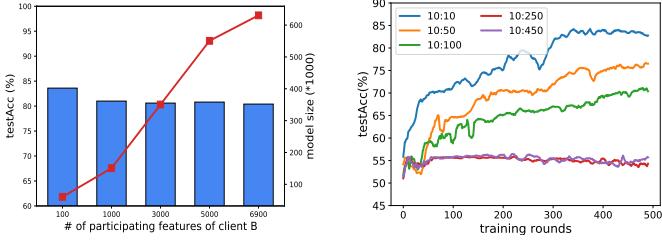
A. Basic Setup of VFL

There are two types of entities involved in VFL: a server S and M clients $\mathcal{M} := \{1, 2, \dots, M\}$. A dataset $U = \{U_1, \dots, U_M\}$ of N samples, $\{X, Y\} := \{x_n, y_n\}_{n=1}^N$, is maintained by the M clients. Let $[N] = \{1, 2, \dots, N\}$. Each client m is associated with a unique set of features $\{f_{m,1}, \dots, f_{m,d_m}\}$, 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 $x_n := [x_{n,1}^\top, x_{n,2}^\top, \dots, x_{n,M}^\top]^\top$. 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 $h_{n,m} := h_m(\theta_m; x_{n,m}) \in \mathbb{R}^{d_m}$ with $d_m \ll d_m$. The server S learns the prediction \hat{y}_n parameterized by the top model $\theta_0 := \{w_1, \dots, w_M, \alpha_0\} \in \Theta$, $w_m \in \mathbb{R}^{d_m}$, $m \in [M]$, where $\{w_1, \dots, w_M\}$ are parameters of the interactive layer which concatenates embedding vectors $h_{n,1}, \dots, h_{n,M}$ in a weighted manner. α_0 denotes the parameters of the succeeding layers of the top model connected to the interactive layer. Ideally, the objective of VFL is to minimize,

$$R(\theta) := \mathbb{E}_{X,Y} L(h(\theta_0, h_{n,1}, \dots, h_{n,M}); y_n) \quad (1)$$

with $h_{n,m} := h_m(\theta_m; x_{n,m})$, $m \in [M]$



(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.

where $\theta := \{\theta_i\}_{i=0}^M$ denotes the global model, which consists of M local models $\theta_1, \dots, \theta_M$ 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 uploads 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.

B. Motivating Examples

Here, we perform data driven analysis to demonstrate the necessity of feature selection in VFL. We illustrate this from two aspects: 1) many clients may possess a large number of redundant features, which results in a low quality and very complex global model; and 2) some clients can possess noisy or task irrelevant features which reduce global model performance. Specifically, we use datasets ARCENE [30] and MADELON [31] as training data to investigate the two observations. 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 [25] based on these two datasets via VFL.

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 the VFL network. 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 the VFL models. 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). The result shows that as the number of noisy features increases, the

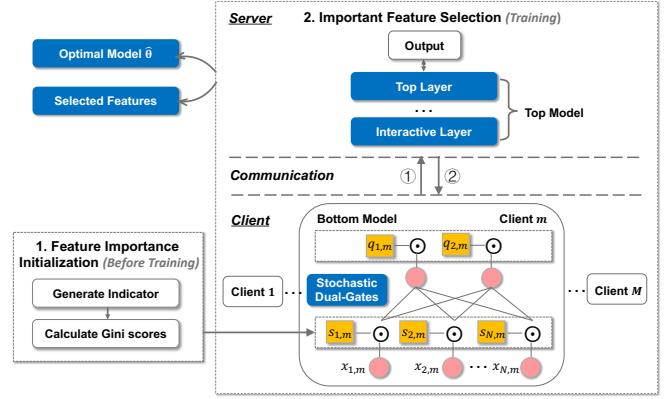


Fig. 2. System overview of FedSDG-FS. ① Send encrypted embeddings, ② send encrypted gradients.

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.

C. Problem Formulation

In a typical VFL system, under the coordination of the server S , all participants train a global model by transferring their local embedding vectors trained using their local datasets. Additionally, we consider a situation in practice in which 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 features. The goal of feature selection in VFL is to simultaneously select a subset of features, and construct a global model $\hat{\theta}$ with the objective by minimizing the risk,

$$R(\theta, s) := \mathbb{E}_{X,Y} L(h(\theta_0, h_{n,1}, \dots, h_{n,M}); y_n) \quad (2)$$

with $h_{n,m} := h_m(\theta_m; x_{n,m} \odot s_m)$, $m \in [M]$,

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. We assume that all participants are semi-honest. They follow the exact protocol of VFL and feature selection, but are curious about others' private information.

IV. THE PROPOSED FedSDG-FS APPROACH

In this section, we first present the system architecture of FedSDG-FS. Then, we illustrate the key technique to enable feature selection to be performed jointly with model training under VFL settings. Finally, we present the details of the FedSDG-FS algorithms.

A. System Overview

FedSDG-FS consists of two modules (as shown in Fig. 2):

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 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, to fulfil the requirement that neither data nor labels can be obtained or inferred by any other party other than their original owners, we propose a secure FS 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 determines the selected features and produces an optimal global model $\hat{\theta}$ with higher accuracy and fast convergence.

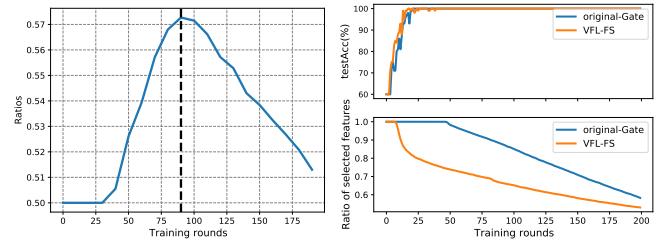
B. Stochastic Dual-Gates for VFL

To achieve accurate feature selection while simultaneously training the global model in VFL, we need to dynamically quantify the influence of features on the global model during training, and increase the probability of selection for highly influential features. In VFL, local embedding vectors are transferred to the server, where the size of embedding vectors 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 embedding vectors being selected. We re-express Eq. (2) into minimizing the l_0 constrained risk:

$$R(\theta, s, q) := \mathbb{E}_{X,Y} L(h(\theta_0, g_{n,1}, \dots, g_{n,M}); y_n) + \lambda \sum_m (|s_m|_0 + |q_m|_0) \quad (3)$$

where $h_{n,m} := h_m(\theta_m; x_{n,m} \odot s_m)$, $g_{n,m} = h_{n,m} \odot q_m$, $q_m = \{0, 1\}^{d_m}$ 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. λ is a weighting factor for the regularization. The l_0 norm penalizes the number of non-zero entries in the vectors s_m, q_m , thus encourages 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 [23].

However, as the optimization of hard feature selection with binary masks suffers from high variance, 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. (3) is simply the sum of the probabilities that $\sum_{i \in [d_m]} P(s_{m,i} > 0) + \sum_{j \in [d_m]} P(q_{m,j} > 0)$, and can be



(a) Ratio of the same selected features by stochastic gate and Gini method
(b) Comparison of the original gate impurity

Fig. 3. Example motivation of FedSDG-FS design. The vertical neural network is trained on the dataset MADEON.

calculated by $\sum_{i \in [d_m]} \Phi\left(\frac{\mu_{m,i}}{\sigma}\right) + \sum_{j \in [d_m]} \Phi\left(\frac{\omega_{m,j}}{\sigma}\right)$, where $\Phi(\cdot)$ is the cumulative distribution function (CDF) of the standard Gaussian distribution. By employing the continuous distribution, we can thus transform Eq. (3) into the following:

$$R(\theta, \mu, \omega) := \mathbb{E}_{X,Y} L(h(\theta_0, g_{n,1}, \dots, g_{n,M}); y_n) + \lambda \left(\sum_{m,i} \Phi\left(\frac{\mu_{m,i}}{\sigma}\right) + \sum_{m,j} \Phi\left(\frac{\omega_{m,j}}{\sigma}\right) \right). \quad (4)$$

To optimize the objective of Eq. (4), 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 performs the differentiation using chain rules [32] based on the Monte Carlo sampling gradient estimator, e.g., for μ_m :

$$\frac{1}{C} \sum_{i \in [C]} \left[\frac{\partial L_n}{\partial g_{n,m}} \cdot \frac{\partial g_{n,m}}{\partial s_m} \cdot \frac{\partial s_{m,i}}{\partial \mu_{m,i}} \right] + \lambda \frac{\partial}{\partial \mu_m} \Phi\left(\frac{\mu_m}{\sigma}\right) \quad (5)$$

where C is the number of Monte Carlo samples. The calculation of gradient of estimator for ω_m is similar to Eq. (5). Thus, we can update μ_m, ω_m via stochastic gradient descent.

Updating the parameters and conducting the above operations require access to all local training samples or training process, which are, however, obfuscated from any third party including the server. In addition, directly applying the stochastic gates to the clients' inputs would require a large number of parameters to be trained (e.g., $\mu_m, m \in [M]$), which slows down the convergence of the global model, and incurs significant computation and communication overhead, especially for high-dimension features.

To address this challenge, we propose an efficient and secure feature selection framework, FedSDG-FS, which leverages Gini impurity for VFL to initialize the importance of individual features to facilitate feature selection. Then important features and significant local embeddings can be selected by the proposed stochastic dual gates, enhanced with PHE and the randomized noisy mechanism for privacy preservation. To illustrate the motivation of the importance initialization, we make the following empirical observations. Firstly, as illustrated in Fig. 3(a), there can be a large ratio of the same features being selected by the Gini impurity [14] and by the stochastic gates in some training rounds. Secondly, Gini impurity initialization can speed up feature selection (Fig. 3(b)). Moreover, the reason that Gini impurity cannot

Algorithm 1: Feature Importance Initialization for VFL

Input : Server S , clients m
Output: Initialized feature importance

- 1 **Server S**
- 2 Generate an indicator matrix A , $\llbracket A \rrbracket \leftarrow Enc(A)$
- 3 Send $\llbracket A \rrbracket$ to all clients
- 4 **Client m**
- 5 Induce a partition $U_{m,1} \cup U_{m,2} \cup \dots \cup U_{m,b}$ of U_m
- 6 Calculate $\llbracket p_{m,k} \rrbracket \leftarrow \sum_{a \in I(U_{m,i})} \llbracket A \rrbracket_{a,k} / |U_{m,i}|$
- 7 Calculate $\llbracket p_{m,k} \rrbracket^2$ with the protocol in [33]
- 8 $\llbracket G(U_{m,i}) \rrbracket \leftarrow 1 - \sum_{k \in [c]} \llbracket p_{m,k} \rrbracket^2$
- 9 $\llbracket G(f_{m,j}) \rrbracket \leftarrow \sum_{i=1}^c \frac{|U_{m,i}|}{|U_m|} \cdot \llbracket G(U_{m,i}) \rrbracket$
- 10 Send $\llbracket G(f_{m,j}) \rrbracket$, $j \in [d_m]$ to the server
- 11 **Server S**
- 12 $G(f_{m,j}) \leftarrow Dec(\llbracket G(f_{m,j}) \rrbracket)$
- 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 $\mu_{m,j}, j \in [d_m]$

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 by FedSDG-FS 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 secure feature selection as well as construct the global VFL model with high inference accuracy and fast convergence.

C. Feature Importance Initialization

M clients have a set $U = \{U_1, \dots, U_M\}$ of N samples, and the corresponding c class labels are stored at the server. 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 $U_{m,1} \cup \dots \cup U_{m,b}$ 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 $U_{m,i}$ belonging to the k -th class. The Gini score of feature $f_{m,j}$ is calculated as $G(f_{m,j}) = \sum_{i \in [b]} \frac{|U_{m,i}|}{|U_m|} \cdot G(U_{m,i})$, 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.

Specifically, the server first generates an indicator matrix A with a size of $N \times c$, where $A_{n,k} = 1$ indicates the category of the n -th sample is k ; otherwise, $A_{n,k} = 0$. Then, the probability $p_{m,k}$ can be calculated as $p_{m,k} = \sum_{a \in I(U_{m,i})} A_{a,k} / |U_{m,i}|$ for client m , where $I(U_{m,i})$ denotes the index set of instances from $U_{m,i}$. To prevent the private label information from being leaked, the server encrypts the matrix A , and sends $\llbracket A \rrbracket$ to all clients. Then, client m calculates the probability $\llbracket p_{m,k} \rrbracket = \sum_{a \in I(U_{m,i})} \llbracket A \rrbracket_{a,k} / |U_{m,i}|$ and uses the protocol in [33] to compute the square of $\llbracket p_{m,k} \rrbracket$ as follows. Firstly, client m generates a random value r and computes $\llbracket u_{m,k} \rrbracket = \llbracket p_{m,k} + r \rrbracket$, such that $p_{m,k}^2$ equals $u_{m,k}^2 - 2u_{m,k} \cdot r + r^2$ and $\llbracket -2u_{m,k} \cdot r + r^2 \rrbracket$ can be locally computed by the client. Then, client m sends $\llbracket u_{m,k} \rrbracket$ to the server. The server decrypts it, computes and sends $\llbracket u_{m,k}^2 \rrbracket$ to client m . Finally, the client computes $\llbracket p^2 \rrbracket = \llbracket u_{m,k}^2 - 2u_{m,k} \cdot r + r^2 \rrbracket$. After calculating $\llbracket p^2 \rrbracket$, client m calculates the Gini impurity $\llbracket G(f_{m,j}) \rrbracket$ of feature $f_{m,j}$, and sends them to the server. The server then decrypts them, and assigns larger initial importance values 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. The main steps are shown in Algorithm 1.

Algorithm 2: 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 $\theta_0 := \{\alpha_0, w_1, w_2, \dots, w_M\}$, $\{\theta_i\}_{i=1}^M$, noise ϵ_{acc} , initialize μ_m with Algorithm 1, $\omega_m \in \mathbb{R}^{d_m}$
- 2 **Client m , $m \in [M]$**
- 3 Select datum (or data mini-batch) $x_{n,m}$
- 4 Sample $\rho_{m,i}, \gamma_{m,j} \sim \mathcal{N}(0, \sigma^2)$, $i \in [d_m], j \in [d_m]$
- 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 $R_m = \sum_{i \in [d_m]} \Phi\left(\frac{\mu_{m,i}}{\sigma}\right) + \sum_{j \in [d_m]} \Phi\left(\frac{\omega_{m,j}}{\sigma}\right)$
- 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 $\llbracket g_{n,m} \rrbracket \leftarrow Enc(g_{n,m})$, sends $\llbracket g_{n,m} \rrbracket$ to the server
- 10 **Server S**
- 11 Calculate the noisy weight $\tilde{w}_m \leftarrow w_m + \epsilon_{acc}$, $m \in [M]$
- 12 Compute $\llbracket \tilde{z}_{n,m} \rrbracket \leftarrow \llbracket g_{n,m} \rrbracket \cdot \tilde{w}_m$
- 13 Add random noise $\llbracket \tilde{z}_{n,m} + \epsilon_s \rrbracket \leftarrow \llbracket \tilde{z}_{n,m} \rrbracket + \epsilon_s$
- 14 Send $\llbracket \tilde{z}_{n,m} + \epsilon_s \rrbracket$ to client m
- 15 **Client m , $m \in [M]$**
- 16 $\tilde{z}_{n,m} + \epsilon_s \leftarrow Dec(\llbracket \tilde{z}_{n,m} + \epsilon_s \rrbracket)$
- 17 Remove noise $z_{n,m} + \epsilon_s \leftarrow \tilde{z}_{n,m} + \epsilon_s - \epsilon_{acc} g_{n,m}$
- 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 $L_n \leftarrow L(h(\alpha_0, z_{n,1}, \dots, z_{n,M}); y_n)$
- 22 **Return** the loss L_n

D. Secure Important Feature Selection

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 $\llbracket g_{n,m} \rrbracket = Enc(g_{n,m})$, which is sent to the server.

2) Forward Propagation on the Server: After receiving the encrypted embedding $\llbracket g_{n,m} \rrbracket$, the server calculates the weighted vector $\llbracket z_{n,m} \rrbracket = \llbracket g_{n,m} \rrbracket \odot w_m$, 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 $\llbracket z_{n,m} \rrbracket$ 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 $\llbracket z_{n,m} \rrbracket$, and sends $\llbracket z_{n,m} + \epsilon_s \rrbracket$ to client m . Then, client m decrypts the noisy weighted sum $\llbracket z_{n,m} + \epsilon_s \rrbracket$, and sends $z_{n,m} + \epsilon_s$ to the server. 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 \tilde{w}_m to calculate the weighted vector $\llbracket \tilde{z}_{n,m} \rrbracket \leftarrow \llbracket g_{n,m} \rrbracket \odot \tilde{w}_m$, where $\tilde{w}_m = w_m + \epsilon_{acc}$, ϵ_{acc} is generated by the client. The forward propagation for secure feature selection is shown in Algorithm 2.

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 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: $\llbracket \frac{\partial L_n}{\partial w_m} \rrbracket, \llbracket \frac{\partial \tilde{L}_n}{\partial g_{n,m}} \rrbracket, \llbracket \frac{\partial L_n}{\partial \alpha_0} \rrbracket$. If the server updates $\llbracket w_m \rrbracket$ by $\llbracket w_m \rrbracket = w_m - \eta_m \llbracket \frac{\partial L_n}{\partial w_m} \rrbracket$, this would result in two encrypted quantities in calculating the weighted vector $z_{n,m} = \llbracket w_m \rrbracket \llbracket g_{n,m} \rrbracket$, which is incompatible with PHE. To avoid this, the server needs to send $\llbracket \frac{\partial L_n}{\partial w_m} \rrbracket$ to client m , and receive the decrypted gradient $\frac{\partial L_n}{\partial w_m}$ back. However, sending $\llbracket \frac{\partial L_n}{\partial w_m} \rrbracket$ 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 (see lines 4-10 of Algorithm 3). Note that the noise ϵ_s generated by the server can be removed when the gradient $\frac{\partial \tilde{L}_n}{\partial w_m}$ still contains noise, where $\frac{\partial \tilde{L}_n}{\partial w_m} = \frac{\partial L_n}{\partial w_m} - \frac{\epsilon_m}{\eta_0}$. With $\frac{\partial \tilde{L}_n}{\partial w_m}$, the server updates the weights as $\tilde{w}_m^{t+1} = w_m^t - \eta \left(\frac{\partial L_n}{\partial w_m} + \frac{\epsilon_m}{\eta_0} \right) = w_m^{t+1} + \epsilon_m$.

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 $w_m^{t+1} = \tilde{w}_m^{t+1} - \epsilon_{acc}$. 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 $\tilde{z}_{n,m}$. Similarly, the extra noise should be added to $\frac{\partial \tilde{L}_n}{\partial g_{n,m}}$, and removed before backpropagation by client m . To achieve this, client m needs to send the encrypted noise $\llbracket \epsilon_{acc} \rrbracket$ to the server, and the server calculates the true gradient via $\llbracket \frac{\partial L_n}{\partial g_{n,m}} \rrbracket = \frac{\partial \tilde{L}_n}{\partial g_{n,m}} - \llbracket \epsilon_{acc} \rrbracket \cdot \frac{\partial L_n}{\partial z_{n,m}}$, and sends the encrypted gradient $\llbracket \frac{\partial L_n}{\partial g_{n,m}} \rrbracket$ to the client m .

4) Backward Propagation on Clients: The client m first decrypts the gradient $\llbracket \frac{\partial L_n}{\partial g_{n,m}} \rrbracket$ received from the server. Then, it updates the local model θ_m and the variable μ_m, ω_m . In this way, model update and feature selection can be accomplished simultaneously. The entire secure backpropagation approach is detailed in Algorithm 3.

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

E. Convergence Analysis

We present convergence results for FedSDG-FS 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 here. 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 then formulate the task as selecting Q such that the MI $I(\cdot)$ between X_Q and Y is maximized:

$$\max_Q I(X_Q, Y) \quad s.t. \quad |Q| = k. \quad (6)$$

Then, under the mild assumption that there exists an optimal subset of indices Q^* , the equation above is equivalent to

$$\max_{\pi \leq \tilde{Q}} I(X \odot \tilde{Q}; Y) \quad s.t. \quad \sum_i \mathbb{E} [\tilde{Q}_i] \leq k, \quad (7)$$

where \tilde{Q} are independently sampled from the Bernoulli distribution with parameter π . Then, we can rewrite this constrained optimization problem as a penalty optimization problem, which is the same as Eq. (3):

$$R = \min L(X \odot \tilde{Q}; Y) + \lambda |\tilde{Q}| \quad (8)$$

So far, we have proved the equivalence between the proposed l_0 constrained optimization and the selection of the optimal feature subset. Next, we give the convergence results of the l_0 constrained optimization for feature selection.

Assumption 1. The gradient $\frac{\partial R(\theta)}{\partial \theta_0}$ is K -Lipschitz continuous, and $\frac{\partial R(\theta)}{\partial \theta_m}$, $m \in [M]$ is K_m -Lipschitz continuous.

Theorem 1. Under Assumption 1, and the 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 $\mathcal{O}(1/T)$.

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 $\mathcal{O}\left(\frac{K||\theta_0 - \hat{\theta}||^2}{\epsilon}\right)$.

V. EXPERIMENTAL EVALUATION

A. Experiment Configuration

1) Datasets. We use 9 datasets with 4 types of data: tabular data, images, texts and audios. These include 2 synthetic datasets, MADELON [31] and FRIEDMAN [34]; and 7 real-world datasets, ARCENE [30], BASEHOCK [35], RELATHE [35], PCMAC [35], GISETTE [36], COIL20 [37] and ISOLET [38]. The synthetic datasets are derived from the feature selection challenge [31], 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 [35]. The descriptions of all datasets are listed in Table I. We employ two clients in our settings, where we divide features into two parts randomly for every dataset, and assign each part to clients A and B. The labels are located in the server. For the text, image and audio datasets, we divide the features randomly by rows for the clients.

TABLE I
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
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 [39] on FRIEDMAN, and neural networks for VFL [25] on the other 8 datasets (see Table II). 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 use the Paillier as the PHE method. We use the Adam optimizer, and set learning rate $\eta = 0.03$, batch size $b = 128$, weight factor $\lambda = 0.1$. We test the accuracy of the global model on the hold-out test datasets. We build our VFL models with Flower 0.19.0 [40] and Pytorch 1.8.1 [41]. 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.

TABLE II
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-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

B. Evaluating Gini Impurity for VFL

Firstly, we evaluate the effectiveness of our Gini impurity metric (FedSDG-FS-gini) designed for feature importance initialization in FedSDG-FS by comparing the test accuracy of the global models to the other three filtering based feature selection strategies, SFFS [17], 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,

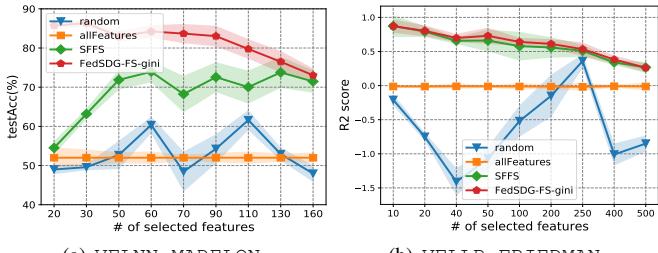


Fig. 4. Test accuracy and R^2 scores vs. number of selected features on synthetic datasets.

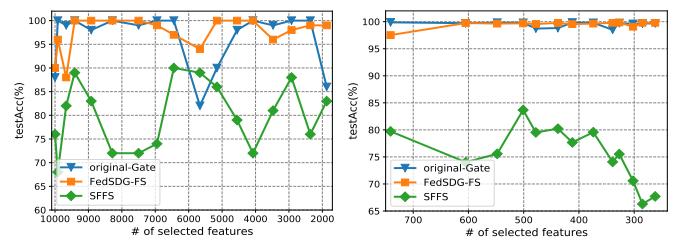


Fig. 5. Test accuracy vs. number of selected features by training models for 20 rounds.

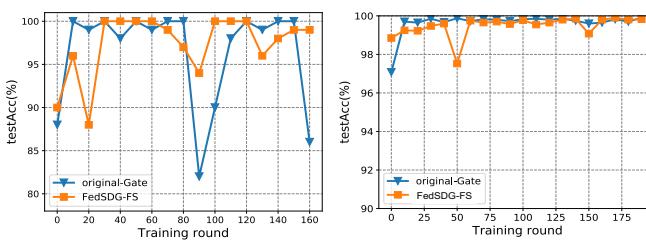


Fig. 6. Test accuracy and number of selected features during model training of original gate selection method and FedSDG-FS.

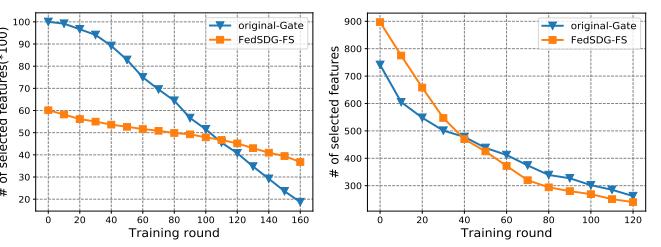
we select features from each dataset in similar proportions and round the numbers of selected features. We perform 5-fold cross validation and report the average R^2 scores, test accuracy and standard deviations in Fig. 4. Here, the R^2 score is defined as $R^2 = 1 - \frac{\sum_{n \in [N]} (y_n - \hat{y}_n)^2}{\sum_{n \in [N]} (y_n - \bar{y})^2}$, 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. Besides, the reason why some of the test accuracy in Fig 4. is less than 50% is that 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.

C. Evaluating Important Feature Selection

After feature importance initialization, FedSDG-FS proceeds to select important features using the stochastic dual-gates. We now evaluate our FedSDG-FS method compared to other baselines, all features participating (allFeatures), SFFS, VFLFS [24], and the original gate based method which has neither gates of the embedding vectors nor importance initialization (original-Gate), using various datasets. For fair comparison, we implement VFLFS [24] without the part that makes use of the non-overlapping samples. Further, we extend a filter feature selection method MS-GINI [14] based on Gini impurity in VFL settings to compare with FedSDG-FS. We perform 5-fold cross validation and report average accuracy.

1) Precision. We use precision to measure the accuracy of 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 7. It can be observed that 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 decreases slightly for the FedSDG-FS and original gate methods, which demonstrates the effectiveness of FedSDG-FS without knowing the number of features to be selected.

2) Learning Accuracy. We compare FedSDG-FS with others 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 III. It can be observed that **FedSDG-FS achieves the highest test accuracy using the fewest features in almost all datasets**. Taking MADELON as an example, the average test accuracy of FedSDG-FS is 0.3%, 33.6%, 27.0%, 47.2%, 50.2% higher than the four methods; while the ratio of selected features is 0.02, 0.97, 0.47, 0.47, 0.47 less than them. In some cases where there are small number of noisy features, and having little negative impact on the model, using all features results the higher accuracy.



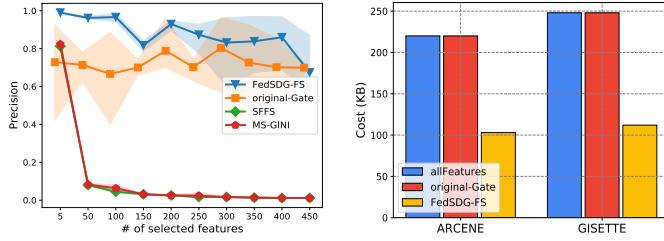


Fig. 7. Precision of different methods on MADELON.

Nevertheless, FedSDG-FS can still achieve comparable test accuracy with fewer features. To further validate the proposed methods, we conducted experiments with five clients and ten clients. Two example results are presented in Table IV, which show that 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.

TABLE III
TEST ACCURACY OF MODELS TRAINED WITH FEATURES SELECTED.

Datasets	Test Accuracy (%) / Ratio of Selected Features					
	allFeatures	SFFS	MS-GINI	VFLFS	original-Gate	FedSDG-FS
MADELON	52.0/ 1.0	65.6/ 0.5	72.2/ 0.5	51.0/ 0.5	98.9/ 0.05	99.2/ 0.03
ARCENE	80.1/ 1.0	95.0/ 0.5	87.5/ 0.5	70.1/ 0.5	97.4/ 1.0	99.8/ 0.58
BASEHOCK	99.7/ 1.0	99.1/ 0.5	98.5/ 0.5	94.4/ 0.5	99.5/ 0.48	99.9/ 0.3
RELATHE	95.5/ 1.0	87.2/ 0.5	92.1/ 0.5	86.5/ 0.5	99.7/ 0.71	99.8/ 0.41
PCMAG	97.6/ 1.0	79.3/ 0.5	90.2/ 0.5	86.1/ 0.5	99.1/ 0.66	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	99.5/ 0.53
COIL20	96.4/ 1.0	65.6/ 0.5	94.8/ 0.5	72.2/ 0.5	91.2/ 1.0	97.5/ 0.71
ISOLET	98.0/ 1.0	92.7/ 0.5	91.4/ 0.5	71.4/ 0.5	93.2/ 1.0	96.7/ 0.75

TABLE IV
TEST ACCURACY OF MODELS TRAINED WITH FEATURES SELECTED.

Datasets	Test Accuracy (%) / Ratio of Selected Features					
	5 Clients					
	allFeatures	SFFS	MS-GINI	VFLFS	original-Gate	FedSDG-FS
ARCENE	85.8/ 1.0	92.2/ 0.5	92.0/ 0.5	71.0/ 0.5	98.2/ 1.0	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	99.5/ 0.45
	allFeatures	SFFS	MS-GINI	VFLFS	original-Gate	FedSDG-FS
ARCENE	91.0/ 1.0	94.0/ 0.5	92.7/ 0.5	82.0/ 0.5	98.6/ 1.0	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	99.8/ 0.44

3) Stability. We evaluate the stability of FedSDG-FS from two aspects, 1) test accuracy of the global model with different numbers of selected features, and 2) test accuracy at different training rounds. 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. 5. The results show that compared to SFFS, the original based method and FedSDG-FS both achieve much higher test accuracy. The performance of FedSDG-FS has little variation in all cases. Then, we calculate the test accuracy of the two models in different training rounds, and plot them in Fig. 6(a) and Fig. 6(b). The results show that FedSDG-FS and the original gate method achieve comparably high test accuracies at different training rounds, while FedSDG-FS is more stable (i.e., the test accuracy of global model drops 2.8% in the 80-th round for FedSDG-FS and 17.9% for the original gate method). The analysis results of test accuracy on other models with different numbers of selected features and at different training rounds are similar to that of VFLNN-ARCENE, VFLNN-GISETTE.

4) Efficiency. Finally, we evaluate the efficiency of FedSDG-FS from two aspects: 1) the speed of the feature importance initialization, and 2) communication saving during model prediction. Firstly, we calculate the number of selected features in different rounds of training VFLNN-ARCENE and VFLNN-GISETTE (Fig. 6(c) and Fig. 6(d)). The results show that with importance initialization, the models can quickly filter out noisy features and select important ones, thus speeding up model training. Secondly, we compare the prediction communication overhead of those models of FedSDG-FS, allFeatures and the original gate method. Fig. 8 shows the average communication cost of each method to select features. The communication cost of FedSDG-FS is more than 50% lower than that of the other methods (e.g., 53.2%, 54.7% lower for datasets ARCENE and GISETTE). The efficiency analysis clearly demonstrated the advantages of the feature importance initialization module of FedSDG-FS.

VI. CONCLUSIONS

In this work, we proposed an efficient and secure 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. Then, we incorporated PHE and randomized noise mechanism into 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 FedSDG-FS significantly outperforms existing approaches in terms of achieving more accurate selection of high-quality features and building global models with better performance. FedSDG-FS achieves the privacy protection goal, e.g., during the entire feature selection and model training process, neither data nor labels will be acquired or inferred by any party other than their original owners.

ACKNOWLEDGMENTS

Han Yu is the corresponding author. 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; NTU Nanyang Assistant Professorship, Future Communications Research & Development Programme (FCP-NTU-RG-2021-014), 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,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207, 2019.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] Y. Liu, Y. Kang, L. Li, X. Zhang, Y. Cheng, T. Chen, M. Hong, and Q. Yang, “A communication efficient vertical federated learning framework,” *Unknown Journal*, 2019.
- [10] T. Chen, X. Jin, Y. Sun, and W. Yin, “Vafl: a method of vertical asynchronous federated learning,” *arXiv preprint arXiv:2007.06081*, 2020.
- [11] 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.
- [12] PowerFL, “Angel powerfl,” <https://data.qq.com/powerfl/>.
- [13] FATE, “Fate-federated-ai,” <https://github.com/FederatedAI/DOC-CHN>.
- [14] X. Li, R. Dowsley, and M. De Cock, “Privacy-preserving feature selection with secure multiparty computation,” *ICML* 2021, 2021.
- [15] 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.
- [16] J. Chen, M. Stern, M. J. Wainwright, and M. I. Jordan, “Kernel feature selection via conditional covariance minimization,” *NeurIPS 2017*, 2017.
- [17] F. Pan, D. Meng, Y. Zhang, H. Li, and X. Li, “Secure federated feature selection for cross-feature federated learning,” 2020.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] C. Hans, “Bayesian lasso regression,” *Biometrika*, vol. 96, no. 4, pp. 835–845, 2009.
- [24] C. Louizos, M. Welling, and D. P. Kingma, “Learning sparse neural networks through l_0 regularization,” *arXiv preprint arXiv:1712.01312*, 2017.
- [25] Y. Zhang and H. Zhu, “Additively homomorphical encryption based deep neural network for asymmetrically collaborative machine learning,” *arXiv preprint arXiv:2007.06849*, 2020.
- [26] 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.
- [27] L. Song, A. Smola, A. Gretton, K. M. Borgwardt, and J. Bedo, “Supervised feature selection via dependence estimation,” in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 823–830.
- [28] G. I. Allen, “Automatic feature selection via weighted kernels and regularization,” *Journal of Computational and Graphical Statistics*, vol. 22, no. 2, pp. 284–299, 2013.
- [29] S. Feng, “Vertical federated learning-based feature selection with non-overlapping sample utilization,” *Expert Systems with Applications*, p. 118097, 2022.
- [30] J. Thomas, “Mass spectrometric data.” [Online]. Available: <https://www.openml.org/d/41157>
- [31] 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.
- [32] A. Miller, N. Foti, A. D’Amour, and R. P. Adams, “Reducing reparameterization gradient variance,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [33] 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.
- [34] J. H. Friedman, “Multivariate adaptive regression splines,” *The annals of statistics*, vol. 19, no. 1, pp. 1–67, 1991.
- [35] A. state university, “Feature selection datasets,” Public online, 2010. [Online]. Available: https://jundongli.github.io/scikit-feature/OLD/datasets_old.html
- [36] U. machine learning repository, “Handwritten digit recognition problem.” [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/Gisette>
- [37] C. University, “Image classification task.” [Online]. Available: <https://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php>
- [38] U. machine learning repository, “Letter-name classification task.” [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/isolet>
- [39] 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.
- [40] Flower, “Flower: A friendly federated learning framework,” Public online, 2022. [Online]. Available: <https://flower.dev/>
- [41] Pytorch, “Pytorch,” Public online, 2022. [Online]. Available: <https://pytorch.org/>