Model Rake: A Defense Against Stealing Attacks in Split Learning

Qinbo Zhang[1], **Xiao Yan**[2*], **Yanfeng Zhao**[1], **Fangcheng Fu**[3], **Quanqing Xu**[4], **Yukai Ding**[1], **Xiaokai Zhou**[1], **Chuang Hu**[1], **Jiawei Jiang**[1*]
[1]School of Computer Science, Wuhan University
[2]Institute for Math & AI, Wuhan University
[3]School of Computer Science, Peking University
[4]OceanBase, Ant Group
{qinbo_zhang, victor_yfzhao, yukai.ding, xiaokaizhou, handc, jiawei.jiang}@whu.edu.cn,
yanxiaosunny@gmail.com, ccchengff@pku.edu.cn, xuquanqing.xqq@oceanbase.com

Abstract

Split learning is a prominent framework for vertical federated learning, where multiple clients collaborate with a central server for model training by exchanging intermediate embeddings. Recently, it is shown that an adversarial server can exploit the intermediate embeddings to train surrogate models to replace the bottom models on the clients (i.e., *model stealing*). The surrogate models can also be used to reconstruct private training data of the clients (i.e., *data stealing*). To defend against these stealing attacks, we propose *Model Rake* (i.e., Rake), which runs two bottom models on each client and differentiates their output spaces to make the two models distinct. Rake hinders the stealing attacks because it is difficult for a surrogate model to approximate two distinct bottom models. We prove that, under some assumptions, the surrogate model converges to the average of the two bottom models and thus will be inaccurate. Extensive experiments show that Rake is much more effective than existing methods in defending against both model and data stealing attacks, and the accuracy of normal model training is not affected.

1 Introduction

Vertical Federated Learning (VFL) [Yang *et al.*, 2019; Zhou *et al.*, 2024; Zhang *et al.*, 2025a] considers the scenario where data is distributed across multiple clients with different feature spaces, and the clients want to train a model collaboratively while preserving data privacy. VFL is widely applied in areas such as healthcare [Tang *et al.*, 2023], finance [Yang *et al.*, 2023a; Zhang *et al.*, 2025b], and Internet of Things (IoT) [Yang *et al.*, 2023b]. For instance, banks can use VFL to train a model to predict customer financial risks without sharing their financial records. Split learning [Vepakomma *et al.*, 2018; Thapa *et al.*, 2022] is arguably the most prevalent framework for VFL due to its efficiency and generality [Gupta and Raskar, 2018; Zhang *et al.*, 2024]. In particular, split learning divides the model into two parts: a bottom model on each client and a top model on the server. Each client processes local features using its bottom model to generate intermediate embeddings, which are sent to the server. The server concatenates these embeddings to train the top model and computes gradients, which are then sent back to the clients for backward propagation.

	Normal	Noisy	Pruning	DPSGD	Rake
ACC (%) ↑	80.41	75.66	78.19	77.96	**80.34**
S-ACC (%) ↓	78.99	72.82	76.49	78.51	**36.87**
AGR (%) ↓	93.08	86.19	88.67	92.79	**34.15**
MSE ↑	0.027	0.048	0.033	0.062	**15.87**

Table 1: Comparing defenses against model and data stealing attacks on the CO dataset. Normal refers to standard training without defense. ACC (higher is better), S-ACC (smaller is better), AGR (smaller is better), and MSE (higher is better) are the performance metrics. We highlight the best defense method for each metric.

However, a recent research [Xu *et al.*, 2024] shows that split learning is prone to both model and data stealing attacks. For model stealing, an adversarial server creates auxiliary data samples to query the system and utilizes the intermediate embeddings generated by the bottom models to train surrogate models, which can replace the original bottom models. The data stealing attack extends the model stealing attack by training a generator model to work with the surrogate model. The generator takes random noise as input to produce fake data samples, which are passed to the surrogate model to produce embeddings. By aligning these fake embeddings and the embeddings of the auxiliary samples, the generator learns to reconstruct data samples. These vulnerabilities highlight the urgent need for practical defense mechanisms to secure split learning against such stealing attacks.

Several defense methods have been proposed [Xu *et al.*, 2024], such as adding Gaussian noise to the embeddings (***Noisy***), pruning element with a small magnitude in the embeddings (***Pruning***), and injecting Gaussian noise to the gradients for the bottom models during each training (***DPSGD***). However, as shown in Table 1, none of these approaches simultaneously meet the three critical requirements for a good defense method: ❶ Transparency: The accuracy of model

*Corresponding authors.

training is not affected, i.e., matching normal training without dense; ❷ Effectiveness: Makes the accuracy of model and data stealing attacks low; ❸ Scalability: Maintains effectiveness when the number of data samples or the model's capacity increases. Table 1 shows that existing methods harm the accuracy of normal training and fail to reduce the accuracy of the stealing attacks. Moreover, we also observe that their effectiveness diminishes with using more data samples or more complex models. Given the shortcomings of existing defense methods, we pose the following research question:

Is it possible to design a defense method against stealing attacks in split that satisfies all three requirements, i.e., transparency, effectiveness, and scalability?

Our Solution: Model Rake. We propose Model Rake (i.e., *Rake*) as a novel defense method to secure split learning against the stealing attacks. The core idea of Rake is to disrupt the adversary's ability to approximate the bottom models by using two bottom models (instead of one) on each client. In particular, each data sample is assigned to a fixed but random bottom model using a hash function, and the sample-model assignment is private for each client. Intuitively, Rake forces the adversary to approximate the two bottom models simultaneously when learning a surrogate model. To make the task difficult for the adversary, we make the two models distinct by differentiating their output spaces. Specifically, the output embeddings of the same model are encouraged to be similar (i.e., intra-model alignment), while the output embeddings of different models are encouraged to be dissimilar (i.e., inter-model contrast). Training works as the normal split learning procedure, and the embedding constraints are only applied locally on each client. Technically, with some assumptions, we prove that Rake forces the surrogate model to converge to the average of the two bottom models. As the two models are in high dimension spaces and sufficiently distinct, averaging them will yield poor accuracy for model stealing. Data stealing is also prevented by Rake because it depends on accurate model stealing.

To evaluate Rake, we compare it with four baselines across six datasets. As shown in Table 1, the training accuracy (ACC) of Rake resembles, satisfying the transparency requirement. Rake also outperforms the baselines in degrading the quality of the surrogate model, leading to significant drops in both task accuracy (S-ACC) and output alignment (AGR) between the surrogate pipeline and the original pipeline. Furthermore, the data reconstructed by the data stealing attack shows significant differences from the original data (as measured by MSE). Our ablation study also confirms that Rake exhibits strong scalability, maintaining consistent defense performance when the number of clients, data volume or model capacity increases.

To summarize, we make the following contributions:

- We introduce Model Rake to defend against stealing attacks in split learning, which utilizes two bottom models on each client and makes the bottom models distinct.
- We provide a theoretical analysis to show that Model Rake forces the surrogate model to converge to the average of the two bottom models during stealing attacks.
- We conduct extensive experiments to demonstrate that Model Rake satisfies all three critical requirements for defense: transparency, effectiveness and scalability.

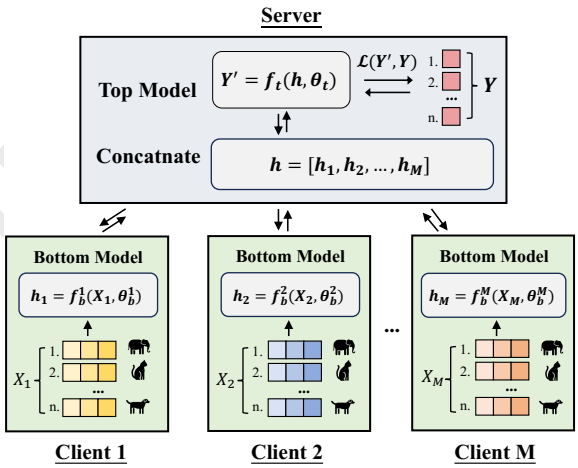

Figure 1: An illustration of the split learning framework.

2 Background

Vertical federated learning and split learning. In split learning framework, as shown in Figure 1, the complete machine learning model is divided between several clients and a central server. Each client is responsible for processing a portion of the sample features, while the server holds all the labels [Xu *et al.*, 2024]. We denote the sample identifiers as $[N] = \{1, \ldots, N\}$ and the set of clients as $[M] = \{1, \ldots, M\}$, with each participant using consistent identifiers for all samples. The feature vector x_i is partitioned across M clients, with each client m holding a subset of the features for all samples, i.e., $\mathbf{X}_m = \{x_i^m \in \mathbb{R}^{F_m} : m \in [M]\}_{i=1}^N$, where F_m is the number of features held by client m and $\sum_{m=1}^M F_m = F$. To enable effective training, the global model $f(\theta)$ is divided into two parts: a bottom model $f_b(\theta_b)$ and a top model $f_t(\theta_t)$. Each client m is responsible for a portion of the bottom model: $f_b^m(\theta_b^m)$, which operates on its local features. The loss function is defined as follows:

$$\mathcal{L}(\mathbf{Y}', \mathbf{Y}) := \mathcal{L}(f_t([h_1, \ldots, h_m, \ldots, h_M], \theta_t), \mathbf{Y}), \quad (1)$$

where $h_m = f_b^m(\mathbf{X}_m, \theta_b^m)$ represents the intermediate embeddings produced by client m's bottom model. The training process for a mini-batch proceeds as follows:

- Clients process local features $\mathbf{X}_m$ using their bottom models $f_b^m(\theta_b^m)$ to generate embeddings h_m.
- The server concatenates the forward embeddings $h = [h_1, \ldots, h_M]$, and processes them with the top model $f_t(h, \theta_t)$ to generate the final output predictions $\mathbf{Y}'$.
- The server then calculates the loss $\mathcal{L}(\mathbf{Y}', \mathbf{Y})$ with $\mathbf{Y}$ (i.e., $\{y_i\}_{i=1}^N$) and generates the gradient $\nabla_{\mathbf{Y}'}\mathcal{L}(\mathbf{Y}', \mathbf{Y})$.
- The server updates the top model with the gradient and computes the gradients $\nabla_{h_m}\mathcal{L}(\mathbf{Y}', \mathbf{Y})$ for each client to update their local models.

	Transparency	Effectiveness	Scalability
Noisy	▭	▭	▭
Pruning	▭	▭	▭
DPSGD	▭	▭	▭
Rake (Ours)	▬	▬	▬

Table 2: Requirements addressed by the defense methods. We use ▬ to show the degree to which each method satisfies the requirements, where ▭ means do not satisfy while ▬ means fully satisfy.

Model and data stealing attacks. Model stealing attack in split learning focuses on replicating the transformation function of the bottom model. In this context, we assume an honest-but-curious server as the adversary. By querying the system with auxiliary samples $\mathbf{X}_m^{aux}$ during the testing phase, the adversary collects the corresponding feature embeddings $f_b^m(\mathbf{X}_m^{aux}, \theta_b^m)$ from each client m. The adversary then uses these pairs of inputs and embeddings $(\mathbf{X}_m^{aux}, h_m^{aux})$ to train a surrogate model $\hat{f}_b^m(\hat{\theta}_b^m)$, aiming to approximate the bottom model's transformation function. This is achieved by minimizing the ℓ_2 distance between the surrogate model's output and the actual embeddings produced by the bottom model:

$$\arg\min_{\hat{\theta}_b^m} \left\| \hat{f}_b^m(\mathbf{X}_m^{aux}, \hat{\theta}_b^m) - f_b^m(\mathbf{X}_m^{aux}, \theta_b^m) \right\|_2. \quad (2)$$

Once the surrogate model $\hat{f}_b^m(\hat{\theta}_b^m)$ is trained, it can replicate the bottom model's functionality. When combined with the server's top model, the adversary can effectively recreate the entire split learning pipeline.

To facilitate data stealing attack, a generator model f_G is introduced. The generator takes random noise $\mathbf{Z}$ as input and produces vectors $f_G(\mathbf{Z})$ with the same dimensionality as the training samples. The adversary then passes $f_G(\mathbf{Z})$ through the surrogate bottom model to obtain the generated embeddings $\hat{f}_b^m(f_G(\mathbf{Z}), \hat{\theta}_b^m)$. The objective is to minimize the ℓ_2 distance between these generated embeddings and the true embeddings $f_b^m(\mathbf{X}_m, \theta_b^m)$ of the original samples $\mathbf{X}_m$:

$$\arg\min_{\theta_G} \left\| \hat{f}_b^m(f_G(\mathbf{Z}, \theta_G), \hat{\theta}_b^m) - f_b^m(\mathbf{X}_m, \theta_b^m) \right\|_2. \quad (3)$$

By optimizing this loss function, the generator produces reconstructed samples $\hat{\mathbf{X}}_m = f_G(\mathbf{Z}, \theta_G)$, which are approximations of the original samples, as their output embeddings closely align with those of the original samples.

Defense methods and requirements. Effective defense methods against stealing attacks in split learning must balance transparency, effectiveness, and scalability. Transparency ensures high training accuracy with minimal interference in normal workflows. Effectiveness focuses on significantly reducing the adversary's ability to exploit intermediate embeddings for model and data stealing. Scalability ensures the defense remains robust as data volume or model capacity increases. There are several existing methods: Noisy, which injects Gaussian noise into feature embeddings; Pruning, which removes the smallest elements from feature embeddings; and DPSGD, which adds Gaussian noise to bottom model gradients during training. However, as shown in Table 2, our experiments demonstrate that none of these methods effectively balance all three requirements. In contrast, our proposed Model Rake (Rake) satisfies all three criteria.

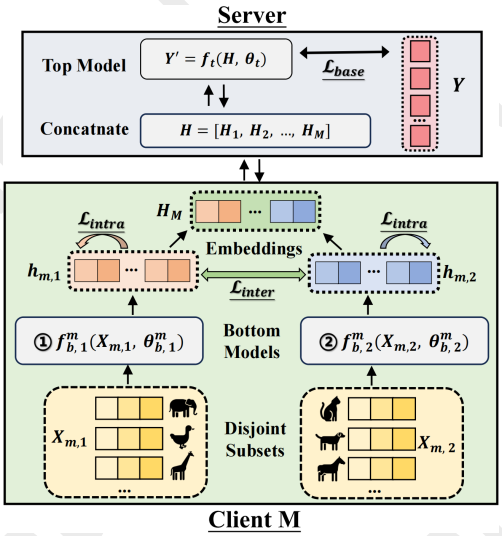

Figure 2: An illustration of our Model Rake. We only show one client for simplicity, and all clients adopt the same defense.

3 Model Rake Framework

In this section, we propose the Model Rake framework to defend against stealing attacks in split learning. As shown in Figure 2, each client is assigned two bottom models, with samples distributed between them using a fixed hash function during training. To further differentiate the output embedding space, we introduce two extra losses: minimizing cosine similarity between embeddings from the two models to create distinct output spaces and maximizing cosine similarity within each model's embeddings to strengthen internal consistency. This setup forces the adversary's surrogate model to approximate both models' transformation functions simultaneously, increasing the complexity of attacks.

3.1 Defense Design

In split learning, stealing attacks exploit the mapping between inputs and intermediate embeddings through auxiliary queries. Model stealing replicates the client's bottom model using this mapping, while data stealing extends it by employing the surrogate model to link a generator's output to true embeddings, enabling precise reconstruction of the original feature space. Therefore, the core defense strategy is to disrupt the adversary's ability to learn this mapping.

Bottom model setup. The training process of the surrogate model requires the adversary to approximate the transformation function of the bottom model. In a typical setting, each client has a single bottom model that generates feature embeddings within a unified space. From a defense perspective, an effective strategy is to increase complexity by diversifying the feature spaces the surrogate model must learn. To achieve this, we draw inspiration from the classic principle

of *The Power of Two Random Choices* [Sitaraman, 2001] in the field of databases and assign two distinct bottom models $f^m_{b,1}(\theta^m_{b,1})$, $f^m_{b,2}(\theta^m_{b,2})$ to each client m.

Sample distribution. To facilitate training, we ensure a consistent and fixed mapping of client samples to the two bottom models. Specifically, the local dataset $\mathbf{X}_m$ for client m is randomly divided into two disjoint subsets $\mathbf{X}_{m,1} = \{x^m_i | i \in I_1\}$, $\mathbf{X}_{m,2} = \{x^m_i | i \in I_2\}$, where $I_1 \cap I_2 = \emptyset$ and $I_1 \cup I_2 = [N]$. This mapping is determined by a client-specific hash function $h(x)$, which assigns each sample x^m_i to one of the subsets, ensuring the allocation remains consistent across all training iterations.

Basic training. Each subset is processed by the corresponding bottom model to generate intermediate embeddings: $h_{m,1} = f^m_{b,1}(\mathbf{X}_{m,1}, \theta^m_{b,1})$, $h_{m,2} = f^m_{b,2}(\mathbf{X}_{m,1}, \theta^m_{b,2})$. These embeddings are then combined and rearranged as H_m, ensuring alignment with the original ordering of $\mathbf{X}_m$, as split learning relies on maintaining this order for accurate server-side processing. Based on that, the basic training loss function $\mathcal{L}_{base}$ for our Model Rake framework under split learning routine can be written as:

$$\mathcal{L}_{base} = \mathcal{L}(f_t([H_1, \ldots, H_m, \ldots, H_M], \theta_t), \mathbf{Y}) \quad (4)$$

Inter-model contrast. To further diversify the roles of the two bottom models, we introduce an inter model contrast loss during training. The goal is to encourage the two models to generate embeddings in distinct spaces, ensuring their outputs represent different transformations. This distinction increases the difficulty for the adversary's surrogate model to approximate both transformations simultaneously. To implement this, we use cosine similarity to measure the angular relationship between the embeddings generated by the two models, applying a penalty to reduce their similarity. For a mini-batch $\mathcal{B}$ of samples from client m, let $\mathcal{B}_1$ and $\mathcal{B}_2$ be the subsets of $\mathcal{B}$ processed by bottom model 1 and 2, respectively, where $\mathcal{B}_1 \cap \mathcal{B}_2 = \emptyset$ and $\mathcal{B}_1 \cup \mathcal{B}_2 = \mathcal{B}$. The embeddings are normalized to unit vectors, $\{h_i | i \in \mathcal{B}_1\}$ and $\{h_j | j \in \mathcal{B}_2\}$. The inter model contrast loss for client m is defined as:

$$\mathcal{L}_{inter} = \sum_{i \in \mathcal{B}_1} \sum_{j \in \mathcal{B}_2} {h_i}^\intercal h_j, \quad (5)$$

Intra-model alignment. While the inter model contrast loss promotes distinctness between the output spaces of the two bottom models, the intra model alignment loss focuses on maintaining consistency and compactness within each model's output space by maximizing cosine similarity within each set of embeddings. This alignment prevents the model from producing highly variable or scattered embeddings for its assigned sample subset. If the embeddings within a model's output space are too spread out, it could lead to overlapping between the two models' output spaces, thus violating their distinctness. For a mini-batch $\mathcal{B}$ of samples from client m and its disjoint subsets $\mathcal{B}_1$ and $\mathcal{B}_2$, the embeddings are normalized to unit vectors before computing the loss. The intra model alignment loss for client m is defined as:

$$\mathcal{L}_{intra} = -\Big(\sum_{\substack{i,j \in \mathcal{B}_1 \\ i \neq j}} {h_i}^\intercal h_j + \sum_{\substack{i,j \in \mathcal{B}_2 \\ i \neq j}} {h_i}^\intercal h_j \Big) \quad (6)$$

Loss composition. The total loss function for the Model Rake framework is composed of three key components: the base loss, inter model contrast loss, and intra model alignment loss. The base loss ensures that the output of the top model aligns with the true labels, while the inter and intra losses enforce the desired separation between the two bottom models' output spaces. Thus, the final loss function is a weighted sum of the base loss and the inter and intra losses:

$$\mathcal{L}_{total} = \mathcal{L}_{base} + \beta \mathcal{L}_{inter} + \gamma \mathcal{L}_{intra}, \quad (7)$$

where $\beta + \gamma = 1$. By combining these losses, we ensure that the adversary's surrogate model is forced to approximate two distinct transformations, with embeddings that remain consistent within each model's output space. This significantly complicates the task for the adversary to construct an accurate model while preserving the original task's accuracy.

3.2 Theoretical Analysis

Theorem 1. *Let the two bottom models A_1 and A_2 be linear models trained with the loss function $\mathcal{L}_{total}$. Assume the data subsets assigned to the two models have identical covariance matrices: $\Sigma_1 = \Sigma_2$. If an adversary applies the model stealing attack routine in Equation 2, the surrogate model A will converge to the average of the two bottom models.*

Here, we give the theoretical proof of Theorem 1. For convenience, we rewrite the problem setting as follows: ❶ Two bottom models A_1 and A_2, surrogate model A; ❷ Disjoint datasets S_1 and S_2; ❸ The adversary attempts to learn a surrogate model A by minimizing embedding differences:

$$L(A) = \sum_{x \in S_1} \|Ax - A_1 x\|^2 + \sum_{x \in S_2} \|Ax - A_2 x\|^2. \quad (8)$$

Based on these settings, the loss $L(A)$ can be expanded as:

$$\begin{aligned} L(A) = &\sum_{x \in S_1} (x^\intercal A^\intercal A x - 2x^\intercal A^\intercal A_1 x + x^\intercal A_1^\intercal A_1 x) \\ &+ \sum_{x \in S_2} (x^\intercal A^\intercal A x - 2x^\intercal A^\intercal A_2 x + x^\intercal A_2^\intercal A_2 x)\,. \end{aligned} \quad (9)$$

For $x^\intercal A^\intercal A x$ (similar to other components in $L(A)$), by applying the matrix trace property, we have: $\mathrm{tr}(x^\intercal A^\intercal A x) = \mathrm{tr}(A^\intercal A x x^\intercal)$, Let $\Sigma_1 = \sum_{x \in S_1} x x^\intercal$, $\Sigma_2 = \sum_{x \in S_2} x x^\intercal$. Then the loss function $L(A)$ becomes:

$$L(A) = \mathrm{tr}(A^\intercal A(\Sigma_1 + \Sigma_2)) - 2\,\mathrm{tr}(A^\intercal(A_1\Sigma_1 + A_2\Sigma_2)). \quad (10)$$

To find the convergence point, we take the derivative of the loss function with respect to A:

$$\frac{\partial L(A)}{\partial A} = 2A(\Sigma_1 + \Sigma_2) - 2(A_1\Sigma_1 + A_2\Sigma_2), \quad (11)$$

and set the gradient to zero, we have:

$$A(\Sigma_1 + \Sigma_2) = A_1\Sigma_1 + A_2\Sigma_2. \quad (12)$$

Assuming ❶ $(\Sigma_1 + \Sigma_2)$ is invertible and ❷ $\Sigma_1 = \Sigma_2$, we have: $A = \frac{A_1 + A_2}{2}$, which means that by performing embedding matching to align the two models, the resulting surrogate model will converge to the average of the two models.

Dataset	CO	SU	RI	BA	HP	YP
# samples	600K	500K	18K	10K	500k	500k
# features	54	18	11	11	11	90
# classes	7	2	2	2	/	/

Table 3: Statistics of the datasets used in our experiments.

4 Experimental Evaluation

We conduct extensive experiments on diverse datasets stored in Oceanbase [Yang *et al.*, 2023c; Yang *et al.*, 2022].

4.1 Experiment Settings

Datasets. Table 3 provides an overview of the datasets used in our experiments. Among them, CO [Blackard, 1998] is designed for seven-class classification, while SU [Whiteson, 2014], RI [MsSmartyPants, 2021], and BA [Topre, 2022] are tailored for binary classification tasks. For the SU dataset, we randomly selected 500,000 samples from the original dataset. HP [Sleem, 2018] and YP [Bertin-Mahieux, 2011] are regression datasets. Note that, due to page limits, the results of the regression task are provided in the Appendix. We also preprocessed the data by normalizing numeric attributes to the range $[0, 1]$ and converting categorical features into one-hot encoded representations. For each dataset, we partitioned the samples into three subsets: a training set (70%) for model training, an auxiliary set (10%) for conducting stealing attacks, and a test set (20%) for performance evaluation.

Models. For all datasets, the default configuration employs a 3-layer fully connected neural network as the bottom model and a non-linear fully connected network, as described in [Fu *et al.*, 2022], as the top model. The surrogate model adopts the same structure as the bottom model. Models in split learning tend to have lower complexity [Khan *et al.*, 2022; Gu *et al.*, 2021], as both data the model are divided into multiple parts. In real-world scenarios, it is uncommon for separate organizations to possess different portion of images. Moreover, advanced architectures such as CNNs are rarely used because there are no effective methods to perform convolution operations in distributed setting [Yang *et al.*, 2019].

Baselines. Based on [Xu *et al.*, 2024], we compare our proposed Model Rake (Rake) with the following baselines:

- Normal: No countermeasures were used under the standard split learning framework.
- Noisy: It injects Gaussian noise with a mean of 0 and a standard deviation from $1e-4$ to 10 into the feature embeddings generated by the target client.
- Pruning: It removes some smallest elements from the feature embeddings produced by the target client with the pruning ratio from 0.1 to 0.9.
- DPSGD: Each client adds Gaussian noise to its own bottom model gradients at each training round with a mean of 0 and a standard deviation from $1e-4$ to $1e-1$.
- Rake-Reveal: The adversarial server is aware of Rake's dual-model defense mechanism but lacks knowledge of how clients allocate their data. It adopts a two-surrogate-model approach to carry out its attacks.

Evaluation protocol. We deploy a cluster consisting of four clients and one central server. The dataset is equally partitioned into four vertical splits, with each client holding one split. Meanwhile, the server has all the labels of the dataset [Xu *et al.*, 2024]. For all experiments, we set the batch size to 1% of the original dataset size and use cross-entropy as the loss function. We use Adam [Kingma and Ba, 2014] as the optimization protocol with a learning rate of $1e-3$. Each model is trained for 100 epochs to ensure convergence. For the weights of the loss function components, we set $\beta = 0.6$, $\gamma = 0.4$, as this configuration yields effective results.

Metrics. We use accuracy (ACC) to evaluate the performance of the main learning task. For the model stealing attack, we employ two metrics: ❶ Stealing Accuracy (S-ACC): The accuracy of the surrogate pipeline (surrogate model + top model) on the main learning task. ❷ Agreement (AGR): The proportion of test samples for which the surrogate pipeline and the true pipeline (bottom model (s) + top model) trained by split learning produce identical predictions. For the defense mechanism, higher ACC is desirable, while lower S-ACC and AGR indicate better protection against the attack. For data stealing attack, we use the mean square error (MSE) between the original data and the reconstructed data to measure the attack's effectiveness. A higher MSE indicates better data protection ability.

Implementation. We run our experiments on a cluster, where each machine has a Intel-i9 CPU and 24GB memory, and the machines are connected via 10GBps Ethernet. The machine communication is implemented using distributed tools available in PyTorch, specifically torch.distributed. Each machine serves as a client, and one machine serves as the server.

4.2 Main Results

Note that, for all defense methods other than Rake (Noisy, Pruning, and DPSGD), we carefully tuned their parameters to ensure optimal performance on each dataset.

Model stealing attack. The results in Table 4 demonstrate that Rake is the best defense approach, offering superior protection against model stealing attack while maintaining high main learning task performance. First, Rake's accuracy matches the Normal setting, confirming that it has no negative impact on main task performance. At the same time, Rake significantly reduces S-ACC to as low as 36.87% and AGR to as low as 34.15% on the CO dataset, showcasing its strong defensive capabilities. On average, Rake reduces S-ACC by 37.23% and AGR by 48.24% compared to the Normal setting. Importantly, Rake remains effective even when the adversarial server is aware of its defense strategy, demonstrating its robustness. In contrast, other defenses like Noisy, Pruning, and DPSGD either fail to sufficiently reduce S-ACC and AGR or negatively impact the performance of the main task. These results clearly highlight that Rake effectively disrupts the server's ability to learn an accurate surrogate model.

Data stealing attack. As the data stealing attack is an extension of model stealing attack, we evaluate Rake's defense

Method	ACC (%) ↑				S-ACC (%) ↓				AGR (%) ↓			
	CO	SU	RI	BA	CO	SU	RI	BA	CO	SU	RI	BA
Normal	80.41±0.19	79.97±0.05	98.98±0.14	86.72±0.09	78.99±0.11	79.84±0.17	98.72±0.13	86.29±0.14	93.08±0.52	97.67±0.72	99.23±0.24	97.14±1.05
Noisy	75.66±0.12	78.55±0.18	95.16±0.08	84.35±0.11	72.82±0.15	74.65±0.23	94.26±0.17	81.76±0.23	86.19±1.32	87.27±2.58	98.04±0.13	91.85±2.32
Pruning	78.19±0.23	79.78±0.11	94.01±0.15	83.32±0.17	76.49±0.13	79.34±0.21	93.76±0.26	80.95±0.18	88.67±1.02	92.38±1.26	98.31±0.74	93.81±2.21
DPSGD	77.96±0.06	79.47±0.06	94.24±0.09	84.78±0.13	78.51±0.16	78.31±0.19	93.93±0.18	82.13±0.15	92.79±1.06	95.44±1.19	98.94±1.01	97.05±0.94
Rake (Ours)	**80.34±0.09**	**79.57±0.13**	**98.85±0.12**	**86.69±0.08**	**36.87±0.11**	**52.78±0.16**	**53.63±0.14**	**48.66±0.15**	**34.15±0.86**	**58.65±1.43**	**55.07±0.46**	**46.28±1.48**
Rake-Reveal	80.34±0.09	79.57±0.13	98.85±0.12	86.69±0.08	33.32±0.21	51.03±0.16	58.83±0.24	57.02±0.24	30.53±1.76	58.36±2.48	60.27±1.12	53.77±2.98

Table 4: Performance comparison in defending against model stealing attack. ACC (higher is better), S-ACC (smaller is better) and AGR (smaller is better) are adopted as performance metrics. We highlight the best defense method for each dataset.

Method	MSE ↑	
	CO	RI
Normal (10^{-2})	2.7±0.14	6.5±0.18
Noisy (10^{-2})	4.8±0.84	8.6±0.73
Pruning (10^{-2})	3.3±0.32	6.8±0.43
DPSGD (10^{-2})	6.2±0.24	7.3±0.29
Rake (Ours)	**15.87±0.58**	**10.73±0.43**
Rake-reveal	15.41±0.42	9.84±0.35

Table 5: Performance comparison in defending against data stealing attack. MSE (higher is better) is used as the evaluation metric. We highlight the best defense method for each dataset.

against this attack on two representative datasets, CO and RI. As shown in Table 5, Rake significantly reduces reconstruction accuracy, increasing the MSE by 580× on CO compared to the Normal setting, demonstrating strong defense capabilities. In contrast, other defense methods show only marginal improvements, with MSE values close to the Normal setting. Furthermore, the Rake-reveal, which simulates a scenario where the adversary is aware of Rake's defense mechanism, still achieves high MSE values, indicating that Rake remains robust even under more challenging conditions.

4.3 Ablation Study and Micro Results

Here, we select the CO dataset and two of the best-performing baseline defenses Noisy and Pruning for further analysis.

Number of clients. We evaluate the stability of Rake and other methods under varying client numbers. As shown in Figure 3, Rake consistently maintains main task accuracy comparable to the Normal setting across all client numbers. In contrast, Noisy and Pruning show reduced defense effectiveness, with increasing S-ACC and AGR values as the client numbers grow. A potential reason is that fewer features are allocated to each client, simplifying the surrogate model's fitting process. Meanwhile, Rake's defense performance remains stable and unaffected by the increase in client numbers, demonstrating its robustness in dynamic environments.

Number of dataset samples. We also examine how Rake and other methods perform with varying scales of the original dataset. Notably, as the total sample number increases, the number of samples available for model stealing attack also grows. As shown in Figure 4, Rake consistently outperforms Noisy and Pruning on the main learning task. While the defense effectiveness of Noisy and Pruning diminishes as the dataset size increases, Rake's performance remains stable and unaffected by the number of samples. This difference arises because Noisy and Pruning rely on perturbing feature embeddings, and larger datasets provide the surrogate model with more opportunities to learn and counteract these perturbations, reducing their effectiveness.

Depth of bottom models. We test the performance of different methods under varying numbers of bottom model layers. As shown in Figure 5, as layers increase, all methods improve on the main learning task. Noisy and Pruning show an initial rise in S-ACC, which eventually stabilizes, while their AGR values gradually increase, indicating that they struggle to protect deeper bottom models. This occurs because deeper models provide more complex feature representations, making it harder for them to maintain effective perturbations. In contrast, Rake provides consistent defense performance, effectively protecting against attacks even on deeper models.

Different surrogate model structure. We evaluate the defense performance of all methods using a 5-layer surrogate model, with the bottom model as a 3-layer network. As shown in Table 6, all methods show reduced effectiveness against the stronger surrogate. Noisy, Pruning, and DPSGD have AGR values over 95%, indicating limited defense. In contrast, Rake's S-ACC and AGR values increase slightly but stay below 40%, ensuring the bottom model's security.

Effect of inter and intra losses. We evaluate the effect of inter model contrast and intra model alignment losses during the defense process. As shown in Table 7, the defense performance without these losses already reaches a relatively ideal level. By further applying them to the output embedding space, the defense effectiveness is significantly enhanced. The individual impact of each loss can be referred to in the experiment: Different weights of inter and intra losses.

Different weights of inter and intra losses. Here, we analyze the impact of the inter and intra loss weights by varying β from 0 to 1, under the constraint $\beta + \gamma = 1$. As shown in Figure 6, the accuracy of the main task tends to stabilize and is largely unaffected by β. In addition, S-ACC and AGR are lower when β takes a middle value ($\beta = 0.6$) and highest when β is 0 or 1. This not only further demonstrates the effectiveness of the two losses, but also confirms that the role

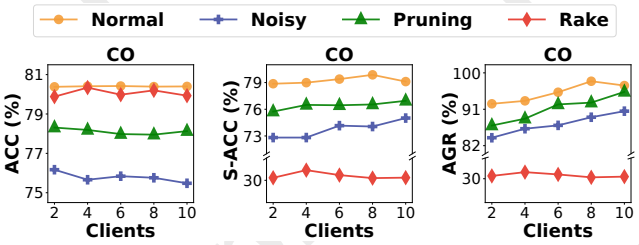

Figure 3: The influence of the number of clients.

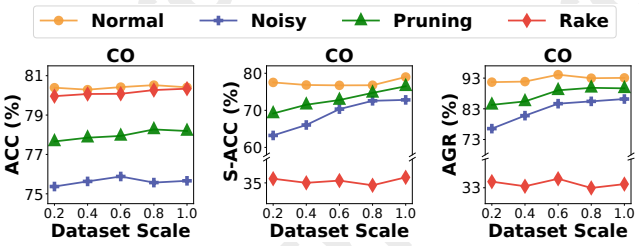

Figure 4: The influence of varying the size of the dataset.

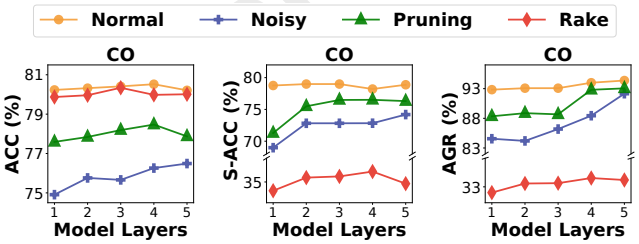

Figure 5: The effect of the number of bottom model layers.

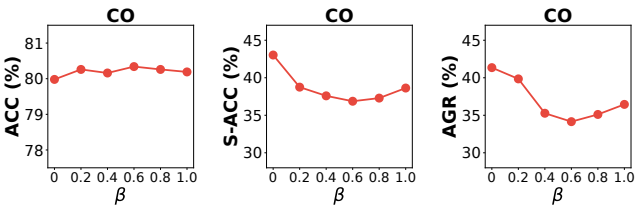

Figure 6: The influence of inter model contrast loss and intra model alignment loss weights by tuning the parameter β.

of either loss cannot be replaced by the other.

Time efficiency. We measured the time to complete the main task training when using no defense method and using Rake as defense. As shown in Table 8, using Rake as the defense method takes only about 8.25% more time than the Normal setting. The reason is that clients need to compute the extra losses and gradients to optimize both two bottom models so as to meet all the requirements.

5 Related Work

The split learning framework introduces privacy and security challenges due to its collaborative nature, where adversaries can exploit shared gradients or feature embeddings to infer sensitive information from other parties. Several studies have examined privacy attacks in split learning. Luo [2021] proposes a feature inference attack, where adversaries exploit publicly available model predictions and apply gradient-based optimization to infer sensitive data. Pasquini [2021] investigates inference attacks, demonstrating how adversaries can recover private data by manipulating the training process and forging gradients. Jin [2021] explores catastrophic data leakage, showing how adversaries can uncover sensitive information by analyzing gradients and model parameters. Xu [2024] highlights the risks of revealing feature embeddings of the bottom model to a malicious server, which can lead to both model and data privacy leaks when combined with appropriate attack methods.

Method	CO		
	ACC (%) ↑	S-ACC (%) ↓	AGR (%) ↓
Normal	80.41	79.46	98.69
Noisy	75.66	75.46	95.12
Pruning	78.19	77.09	98.19
Rake (Ours)	**80.34**	**38.42**	**37.32**
Rake-Reveal	80.34	37.27	37.68

Table 6: Performance in defending against model stealing attack with a 5-layer surrogate model. The best defense is highlighted.

Method	CO		
	ACC (%) ↑	S-ACC (%) ↓	AGR (%) ↓
w/o inter-intra	80.39	59.95	58.24
inter-intra	80.34	36.87	34.15

Table 7: Effect of the inter-model and intra-model losses in defending against model stealing attack under default experiment settings.

Method	Time (s)			
	CO	SU	RI	BA
Normal	851	763	69	47
Rake	917 (+7%)	825 (+8%)	75 (+8%)	52 (+10%)

Table 8: Running time for normal training and using Rake.

6 Conclusions

In this paper, we propose Model Rake as a defense method against model and data stealing attacks in split learning. Model Rake uses a dual model setup on each client with inter model contrast and intra model alignment losses to further differentiate the embedding spaces of the two bottom models. Theoretically, we prove that, under mild assumptions, the surrogate model will converge to the average of the two bottom models. Extensive experiments demonstrate that Model Rake significantly outperforms the baselines in defending stealing attacks and satisfies all key defense requirements.

Acknowledgments

This work was sponsored by Key R&D Program of Hubei Province (2023BAB077), National Natural Science Foundation of China (62472327), and the Fundamental Research Funds for the Central Universities (2042025kf0040). This work was supported by Ant Group through CCF-Ant Research Fund (CCF-AFSG RF20240104) and Sichuan Clinical Research Center for Imaging Medicine (YXYX2402).

References

[Bertin-Mahieux, 2011] T Bertin-Mahieux. YearPrediction-MSD Dataset. UCI Machine Learning Repository, 2011. DOI: https://doi.org/ 10.24432/C50K61.

[Blackard, 1998] Jock Blackard. Covertype. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50K5N.

[Fu *et al.*, 2022] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *31st USENIX security symposium (USENIX Security 22)*, pages 1397–1414, 2022.

[Gu *et al.*, 2021] Bin Gu, An Xu, Zhouyuan Huo, Cheng Deng, and Heng Huang. Privacy-preserving asynchronous vertical federated learning algorithms for multiparty collaborative learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6103–6115, 2021.

[Gupta and Raskar, 2018] Otkrist Gupta and Ramesh Raskar. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116:1–8, 2018.

[Jin *et al.*, 2021] Xiao Jin, Pin-Yu Chen, Chia-Yi Hsu, Chia-Mu Yu, and Tianyi Chen. Cafe: Catastrophic data leakage in vertical federated learning. *Advances in Neural Information Processing Systems*, 34:994–1006, 2021.

[Khan *et al.*, 2022] Afsana Khan, Marijn ten Thij, and Anna Wilbik. Communication-efficient vertical federated learning. *Algorithms*, 15(8):273, 2022.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Luo *et al.*, 2021] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. Feature inference attack on model predictions in vertical federated learning. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 181–192. IEEE, 2021.

[MsSmartyPants, 2021] MsSmartyPants. Rice type binary classification. *https://www.kaggle.com/datasets /mssmartypants/rice-type-classification*, 2021.

[Pasquini *et al.*, 2021] Dario Pasquini, Giuseppe Ateniese, and Massimo Bernaschi. Unleashing the tiger: Inference attacks on split learning. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2113–2129, 2021.

[Sitaraman, 2001] Ramesh Sitaraman. The power of two random choices: A survey of techniques and results. 2001.

[Sleem, 2018] Ahmed Sleem. HousePricing. Kaggle Big Data Competition Platform, 2018. DOI: https://www.kaggle.com/datasets/ greenwing1985/housepricing/data.

[Tang *et al.*, 2023] Fei Tang, Shikai Liang, Guowei Ling, and Jinyong Shan. Ihvfl: a privacy-enhanced intention-hiding vertical federated learning framework for medical data. *Cybersecurity*, 6(1):37, 2023.

[Thapa *et al.*, 2022] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. Splitfed: When federated learning meets split learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8485–8493, 2022.

[Topre, 2022] Gaurav Topre. Bank customer churn dataset. *https://www.kaggle.com/datasets/gauravt opre/bank-customer-churn-dataset*, 2022.

[Vepakomma *et al.*, 2018] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.

[Whiteson, 2014] Daniel Whiteson. SUSY. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C54606.

[Xu *et al.*, 2024] Xiangrui Xu, Wei Wang, Zheng Chen, Bin Wang, Chao Li, Li Duan, Zhen Han, and Yufei Han. Finding the piste: Towards understanding privacy leaks in vertical federated learning systems. *IEEE Transactions on Dependable and Secure Computing*, 2024.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[Yang *et al.*, 2022] Zhenkun Yang, Chuanhui Yang, Fusheng Han, Mingqiang Zhuang, Bing Yang, Zhifeng Yang, Xiaojun Cheng, Yuzhong Zhao, Wenhui Shi, Huafeng Xi, et al. Oceanbase: a 707 million tpmc distributed relational database system. *Proceedings of the VLDB Endowment*, 15(12):3385–3397, 2022.

[Yang *et al.*, 2023a] Liu Yang, Di Chai, Junxue Zhang, Yilun Jin, Leye Wang, Hao Liu, Han Tian, Qian Xu, and Kai Chen. A survey on vertical federated learning: From a layered perspective. *arXiv preprint arXiv:2304.01829*, 2023.

[Yang *et al.*, 2023b] Ruikang Yang, Jianfeng Ma, Junying Zhang, Saru Kumari, Sachin Kumar, and Joel JPC Rodrigues. Practical feature inference attack in vertical federated learning during prediction in artificial internet of things. *IEEE Internet of Things Journal*, 11(1):5–16, 2023.

[Yang *et al.*, 2023c] Zhifeng Yang, Quanqing Xu, Shanyan Gao, Chuanhui Yang, Guoping Wang, Yuzhong Zhao, Fanyu Kong, Hao Liu, Wanhong Wang, and Jinliang Xiao.

Oceanbase paetica: A hybrid shared-nothing/shared-everything database for supporting single machine and distributed cluster. *Proceedings of the VLDB Endowment*, 16(12):3728–3740, 2023.

[Zhang *et al.*, 2024] Qinbo Zhang, Xiao Yan, Yukai Ding, Quanqing Xu, Chuang Hu, Xiaokai Zhou, and Jiawei Jiang. Treecss: An efficient framework for vertical federated learning. In *International Conference on Database Systems for Advanced Applications*, pages 425–441. Springer, 2024.

[Zhang *et al.*, 2025a] Qinbo Zhang, Xiao Yan, Yukai Ding, Fangcheng Fu, Quanqing Xu, Ziyi Li, Chuang Hu, and Jiawei Jiang. Hacore: Efficient coreset construction with locality sensitive hashing for vertical federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 22515–22523, 2025.

[Zhang *et al.*, 2025b] Ziyi Zhang, Hang Yin, Susie Xi Rao, Xiao Yan, Zhurong Wang, Weiming Liang, Yang Zhao, Yinan Shan, Ruixuan Zhang, Yuhao Lin, et al. Identifying e-commerce fraud through user behavior data: Observations and insights. *Data Science and Engineering*, pages 1–16, 2025.

[Zhou *et al.*, 2024] Xiaokai Zhou, Xiao Yan, Xinyan Li, Hao Huang, Quanqing Xu, Qinbo Zhang, Yen Jerome, Zhaohui Cai, and Jiawei Jiang. vfdv-im: An efficient and securely vertical federated data valuation. In *International Conference on Database Systems for Advanced Applications*, pages 409–424. Springer, 2024.