# Vertical Federated Learning Across Second-hop Parties

Zikai Dou(✉) and Fei Chiang

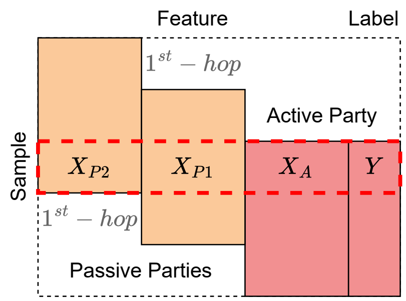
McMaster University, Hamilton, Canada
{douz7, fchiang}@mcmaster.ca

**Abstract.** Vertical federated learning (VFL) enables parties with different features to collaboratively develop models on overlapping samples without directly sharing raw data. Conventional VFL systems typically require data overlap among all parties to perform training and inference. Recent work has relaxed this assumption to allow federated inference on non-overlapping samples. However, current methods struggle to handle scenarios where the active party only shares sample overlap with a subset of passive parties, thereby missing critical feature information from non-overlapping parties. We introduce Vertical Federated Learning Across Second-hop Parties (VFL-ASP), an enhanced VFL framework that improves feature utilization across second-hop passive parties. These parties share sample overlap with first-hop passive parties, which directly overlap with the active party. VFL-ASP extracts hidden embeddings from overlapping samples among the first and second-hop parties under encrypted federation and learns embedding approximations, which are then utilized alongside active party data to construct a VFL system. We apply knowledge distillation to refine a student model with soft labels from the VFL teacher, enabling local processing of non-overlapping data. Our evaluation on three real-world datasets demonstrates that VFL-ASP achieves improved performance over traditional VFL baselines.

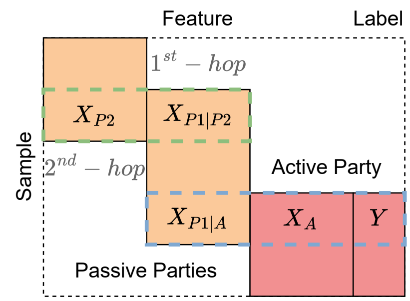
**Keywords:** Vertical federated learning · Second-hop parties · Embedding extraction

## 1 Introduction

Federated learning (FL) provides a decentralized approach for training machine learning models on distributed data without explicitly sharing data across parties, enhancing privacy and security [13,15,16,18,22,29]. FL has been extensively used in many domains, particularly in healthcare, where collaboration across hospitals enables global hypothesis testing and subgroup analyses, surpassing localized hospital settings [3]. Vertical federated learning (VFL) extends FL to vertically partitioned data, allowing multiple parties with distinct features on shared samples to jointly train models without exchanging raw data or model parameters. Each party trains a local model and uploads intermediate results to the global model as input for training. The global model then returns gradients




**(a)** Standard data distribution assumes data overlap among all parties.




**(b)** No data overlap exists between the second-hop passive and active parties.

**Fig. 1.** VFL data overlap configurations.

to update local model parameters, enabling all parties to collaboratively conduct inference afterward [20]. VFL has emerged as a promising approach for privacy-preserving inference in scenarios with shared samples but unique features [27].

Standard VFL systems assume multiple passive parties contribute features while the active party provides both features and labels. In addition, only overlapping data shared across all (passive and active) parties is used for training local and global models, as well as for inference. Figure 1a shows such a shared data overlap (dotted red box) between passive parties $P1, P2$ (denoted in orange), and the active party $A$ (denoted in red).

Unfortunately, the assumption that all passive parties share common data overlap with the active party rarely holds in practice due to data-sharing constraints, heterogeneous sources, and other limitations. Additionally, different data usage policies across parties lead to feature variability. Existing work has provided solutions to allow inference on local non-overlapping data while still requiring overlapping data from all parties for training [10, 17, 24]. These methods fail to leverage critical feature information from (second-hop) passive parties that do not directly overlap with the active party but share sample alignment with intermediate (first-hop) passive parties that overlap with the active party. We adapt the concept of "hop" from graph neural networks (GNNs), where information can propagate through intermediate nodes instead of direct connections. Figure 1b shows such an example setting where the second-hop passive party $P2$ lacks direct overlap with the active party $A$ but is connected through the first-hop passive party $P1$ (dotted green box), which directly overlaps with the active party (dotted blue box). These data overlap structures present challenges for traditional VFL methods, as highlighted in the following example.

*Example 1.* Figure 2 illustrates a clinical hospital data configuration where Hospital $A$ (active party) contains patient demographic information as features and disease outcomes as labels. Hospital $P1$ (first-hop passive party) holds patient

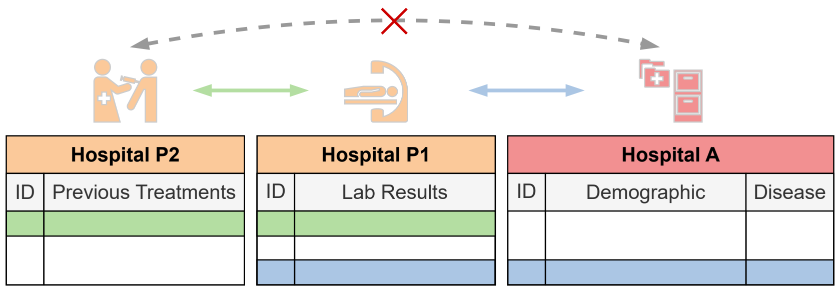



**Fig. 2.** Illustration of second-hop data configuration across hospitals.

lab results, and shares patient overlap with both Hospital $A$ (via blue records) and Hospital $P2$ (via green records). Hospital $P2$ (second-hop passive party) stores data features on previous treatments, which are critical for predicting disease outcomes, but only shares overlapping records with Hospital $P1$.

We propose Vertical Federated Learning Across Second-hop Parties (VFL-ASP), a novel framework that incorporates features from the second-hop parties during learning and inference. VFL-ASP first extracts hidden embeddings under encrypted federation from overlapping data between the first and second-hop parties, and then utilizes auto-encoder [9] to perform semi-supervised learning to generate embedding approximations that capture feature information learned from second-hop parties. Subsequently, the framework leverages these approximations (instead of local data from first-hop) alongside active party data to construct a VFL system. To enhance the capability, VFL-ASP employs knowledge distillation [8], enabling a student model at the active local party to make inferences independently, supervised by the VFL teacher model.

We propose a more efficient approach to federated Singular Value Decomposition (SVD) by concatenating encrypted overlapping data from both hops, enabling hidden embedding extraction with a single encryption matrix. We present formalized details of our approach. In contrast, previous approaches are designed to use the summation of overlapping data with two encryption matrices (left and right orthogonal masks), leading to higher computation costs [1, 10].

We evaluate our model on three real-world datasets with diverse data partitioning strategies to explore varying feature splits and sample overlaps among parties, comparing its performance against standard VFL and local models. The results show that VFL-ASP consistently achieves the highest accuracy, particularly outperforming the standard VFL and local models by $+1.387\%$ and $+5.474\%$ accuracy, respectively. We perform an ablation study to assess the effectiveness of the extraction module and conduct a runtime analysis to evaluate efficiency.

**Contributions.** We make the following contributions:

1. We introduce VFL-ASP, a new framework that integrates feature information from second-hop passive parties into the training and inference process.

**Table 1.** Notation summary.

| Symbol | Description |
|---|---|
| $X_i$, $X_{i\|j}$ | Data matrices for party $i$ and its overlap with party $j$ |
| $X_i^\epsilon$ | Encrypted data for federated SVD |
| $G$, $M_i$ | Global and local models |
| $\phi$, $\theta_i$ | Global and local parameters |
| $Y$ | Ground truth label |
| $\hat{Y}$, $\tilde{Y}$ | Model prediction and soft label |
| $E$, $\hat{E}$ | Decrypted embedding and embedding approximation |

2. We propose a more efficient approach of federated SVD via the concatenation of encrypted overlapping data from the first-hop and second-hop parties to extract hidden embeddings with a single encryption matrix.
3. We assess the performance of VFL-ASP on three real-world datasets with varying feature splits and sample overlaps among parties, complemented by a runtime analysis and an ablation study.

## 2 Overview

We briefly introduce VFL, followed by an overview of VFL-ASP architecture. Table 1 summarizes the notations used in this study.

### 2.1 Vertical Federated Learning

Define $X_i^{Sample}$ and $X_i^{Feature}$ as the sample and feature space corresponding to the matrix's rows and columns, respectively. Without loss of generality, consider two passive parties $P1$ and $P2$, and an active party $A$. The necessary conditions for VFL are: $X_{P2}^{Feature} \neq X_{P1}^{Feature} \neq X_A^{Feature}$, ensuring unique features for each party, and $X_{P2}^{Sample} = X_{P1}^{Sample} = X_A^{Sample}$, indicating shared samples among all parties.

Suppose the collaborative training is based on the overlapping data with $m$ samples across $k$ parties and each party holds $n_i$ features. Then, each party owns a data matrix $X_i \in \mathbb{R}^{m \times n_i}$. Hence, the problem can be formulated as:

$$\min \mathcal{L}_{vfl}(G(_{\phi|M_1(\theta_1|X_1),\ldots,M_k(\theta_k|X_k)}), Y),$$

where the VFL training loss $\mathcal{L}_{vfl}$ is computed from the global model $G$ with ground truth $Y$.

**Training Procedure.** The training procedure is shown in Figure 3. Passive parties typically only communicate with the active party, which acts as the coordinator to control the training and inference process of the global model. However, in certain scenarios, a third party may be involved, responsible for encryption and decryption tasks [21]. Each party will train a local model $M_i$ with its private data $X_i$ first and then send the intermediate outputs $H_i$ to the active party without directly sharing raw data. Next, the active party will

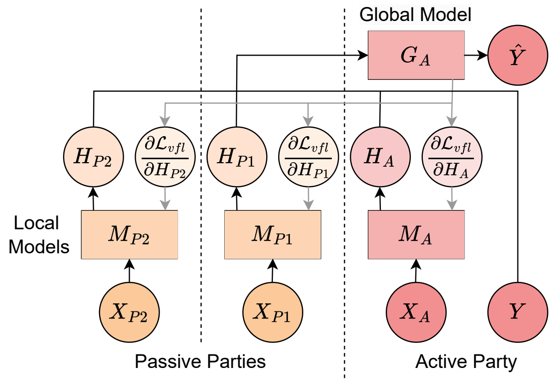



**Fig. 3.** VFL training procedure.

feed these intermediate results into a global model $G$ with ground truth $Y$ to compute the training loss $\mathcal{L}_{vfl}$ for updating the global parameters $\phi$ by the partial derivative $\frac{\partial \mathcal{L}_{vfl}}{\partial \phi}$. Then, the active party will send gradients $\frac{\partial \mathcal{L}_{vfl}}{\partial H_i}$ back to each local model. Finally, the local models will update the local parameters $\theta_i$ until convergence by the following chain rule for the partial derivative:

$$\frac{\partial \mathcal{L}_{vfl}}{\partial \theta_i} = \frac{\partial \mathcal{L}_{vfl}}{\partial H_i} \frac{\partial H_i}{\partial \theta_i}.$$

During inference with new data samples, each party will first generate the intermediate outputs with unique feature information and then the global model will utilize these intermediate results to generate the prediction $\hat{Y}$.

**Privacy and Security.** In a standard VFL system, privacy risks can arise inside and outside. Inside risks stem from vulnerabilities within the system itself. Outside risks originate from threats posed by external attackers. The parties within the system can be labeled as honest, semi-honest, or malicious. A semi-honest attacker obeys the VFL protocol but seeks to access the private data of other parties, whereas a malicious attacker violates the VFL protocol. Privacy-preserving protocols involved in a typical VFL framework are extensively covered in the survey [20]. In this study, we assume all parties are semi-honest, which is widely accepted in the FL system [27]. Although weaker than the malicious setting, this assumption is highly efficient. Under this setting, all parties strictly adhere to the protocol, yet they may be curious about other parties and attempt to obtain additional information from the transferred data.

### 2.2 VFL-ASP Architecture

The architecture of VFL-ASP is illustrated in Figure 4, which proceeds along four main modules. We develop the initial two modules as extensions for VFL systems to leverage feature information from the second-hop passive parties, ensuring easy integration with the existing VFL frameworks.

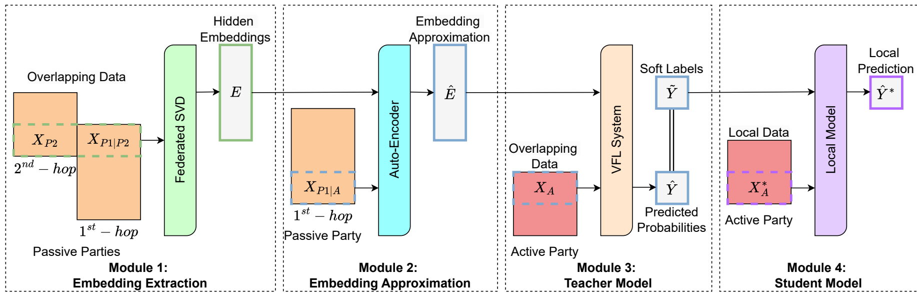



**Fig. 4.** Framework overview of VFL-ASP.

**Embedding Extraction**: We first extract hidden embeddings under encrypted federation from shared overlapping samples using federated SVD.
**Embedding Approximation**: We apply semi-supervised training using an auto-encoder to approximate the embeddings. By utilizing the first two modules, VFL-ASP enables the integration of features from the second-hop.
**Teacher (VFL) Model**: The framework leverages the approximations (instead of local data from first-hop) alongside active party data to construct a VFL system, generating both predictions and soft labels.
**Student Model**: The soft labels reflect the probability distribution of VFL predictions, which supervise a student model at the local active party, allowing independent inference.

Our framework currently reaches second-hop parties, assuming features retain high relevance through shared samples with first-hop parties that directly participate in the VFL system. While VFL-ASP can extend to additional hops, higher-hop features may have weaker correlations with active party labels, and embedding approximations may accumulate errors across multiple hops.

## 3 Methodology

### 3.1 Embedding Extraction

**Federated SVD.** We propose an efficient approach to federated Singular Value Decomposition (SVD) to extract hidden embeddings $E$ from overlapping data $X_{P2}$ and $X_{P1|P2}$, inspired by FedSVD [1]. Original FedSVD aims to achieve lossless accuracy and high efficiency without directly sharing raw data by using matrix masking technique. For a given matrix $X \in \mathbb{R}^{m\times n}$, SVD decomposes it into three matrices:

$$X = U\Sigma V^T, \tag{1}$$

where $U \in \mathbb{R}^{m\times m}$ and $V^T \in \mathbb{R}^{n\times n}$ are orthogonal matrices containing left and right singular vectors, and $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix with singular values in decreasing order. In a typical federated SVD scenario with $k$ parties, each possessing a data matrix $X_i \in \mathbb{R}^{m\times n_i}$, where $m$ represents the number of overlapping samples, $n_i$ is the number of features within each party, and $\sum_{i=1}^{k} n_i = n$. These $k$ parties collectively perform SVD on the concatenated data $[X_1, ..., X_k] = X \in \mathbb{R}^{m \times n}$, yielding:

$$[X_1, ..., X_k] = U\Sigma[V_1^T, ..., V_k^T]. \tag{2}$$

Each party derives the result as $X_i = U\Sigma V_i^T$, where $U \in \mathbb{R}^{m \times m}$ and $\Sigma \in \mathbb{R}^{m \times n}$ are shared among all parties, but $V_i^T \in \mathbb{R}^{n \times n_i}$ can only be accessed by party $i$.

The FedSVD framework implements federated SVD by encrypting the private data by multiplying two random orthogonal matrices on both sides at the beginning, which can be fully removed from the SVD results in the end [1]. While FedSVD discusses both orthogonal matrices $U$ and $V$ in detail, for brevity, we will discuss $U$ only. Assuming $k$ parties are sharing overlapping samples, the overlapping data can be represented as $X = [X_1, ..., X_k]$. To ensure privacy preservation, a trusted third party generates two random orthogonal matrices $P \in \mathbb{R}^{m \times m}$ and $Q^T \in \mathbb{R}^{n \times n}$. $Q^T$ is further split into $k$ matrices $Q^T = [Q_1^T, ..., Q_k^T]$, where $Q_i^T \in \mathbb{R}^{n \times n_i}$. The third party sends $P$ and $Q_i^T$ to each party, which then sends the encrypted data $X_i^\epsilon = PX_iQ_i \in \mathbb{R}^{m \times n}$ to a semi-honest central server. Since we know that

$$PXQ = P[X_1, ..., X_k][Q_1^T, ..., Q_k^T]^T = \sum_{i=1}^{k} PX_iQ_i = \sum_{i=1}^{k} X_i^\epsilon, \tag{3}$$

we can set $PXQ = X^\epsilon$. Therefore, $\sum_{i=1}^{k} X_i^\epsilon = X^\epsilon$. The server then runs the SVD algorithm on the summation of the encrypted data from $k$ parties, resulting in $X^\epsilon = U^\epsilon \Sigma^\epsilon V^{\epsilon T}$. Thus, we obtain

$$X = P^T X^\epsilon Q^T = P^T U^\epsilon \Sigma^\epsilon V^{\epsilon T} Q^T. \tag{4}$$

If we run the SVD algorithm on raw data $X$ directly, we get $X = U\Sigma V^T$. Since orthogonal matrices only represent rotations and reflections, and do not change the magnitude of vectors, the singular values represented by $\Sigma$ and $\Sigma^\epsilon$ are equal. Therefore, $\Sigma = \Sigma^\epsilon$ and

$$[P^T U^\epsilon]\Sigma^\epsilon[V^{\epsilon T} Q^T] = U\Sigma^\epsilon V^T. \tag{5}$$

We may use $P^T U^\epsilon$ to approximate $U$ [1]. Lastly, the server releases the encrypted orthogonal matrix $U^\epsilon$ to the $k$ parties, and each party uses $P^T U^\epsilon$ to approximate $U$.

**Hidden Embeddings.** VFL-ASP utilizes a single random orthogonal matrix $P \in \mathbb{R}^{m \times m}$, generated by a trusted third party and distributed to each passive party. Each party then encrypts its data as $X_i^\epsilon = PX_i \in \mathbb{R}^{m \times n_i}$ and sends it to a semi-honest central server (operated by the active party). Given that

$$PX = P[X_1, ..., X_k] = [PX_1, ..., PX_k] = [X_1^\epsilon, ..., X_k^\epsilon], \tag{6}$$

we can set $PX = X^\epsilon \in \mathbb{R}^{m \times n}$. The server runs the SVD algorithm on the concatenation of the encrypted data matrices instead of their summation, resulting in $X^\epsilon = U^\epsilon \Sigma^\epsilon V^{\epsilon T}$. Thus, we have

$$X = P^T X^\epsilon = P^T U^\epsilon \Sigma^\epsilon V^{\epsilon T}. \tag{7}$$

Similarly, since the SVD result of raw data $X$ is $X = U \Sigma V^T$ and $\Sigma^\epsilon = \Sigma$, then we get

$$[P^T U^\epsilon] \Sigma^\epsilon V^{\epsilon T} = U \Sigma^\epsilon V^T. \tag{8}$$

Finally, the server releases the encrypted orthogonal matrix $U^\epsilon$ and singular value $\Sigma^\epsilon$ back to the $k$ parties. Each party can approximate $U$ with $P^T U^\epsilon$ and approximate $U\Sigma$ using $[P^T U^\epsilon]\Sigma^\epsilon$, which is the decrypted hidden embedding $E$ we want to extract.

Note that directly recovering $X$ from $X^\epsilon$ without knowing $P$ is not possible, even if the server knows the SVD result of $X^\epsilon$. This guarantees data privacy between the passive and active parties. In practical scenarios where $m \gg n$, to reduce computational costs, we trim the encrypted orthogonal matrix $U^\epsilon$ from $\mathbb{R}^{m \times m}$ to $\mathbb{R}^{m \times n}$, preserving only the first $n$ significant columns. $\Sigma^\epsilon$ keeps the singular values and forms a diagonal matrix as $\mathbb{R}^{n \times n}$. Eventually, the first-hop passive party obtains the decrypted embedding matrix $E$ as $[P^T U^\epsilon]\Sigma^\epsilon \in \mathbb{R}^{m \times n}$, which shares the same dimensions as the original overlapping data $X$.

### 3.2 Embedding Approximation

We employ a standard auto-encoder architecture, consisting of an encoder $F_{enc}(\cdot)$ followed by a decoder $F_{dec}(\cdot)$. The input data from the first-hop passive party is fed into an encoder with three hidden layers to compute the embedding approximations $\hat{E}$. We input these predicted embeddings into the decoder to reconstruct the original input data. The model designs the dimensions of each layer in the encoder and then mirrors these dimensions for the decoder, creating a symmetric structure. Specifically, for the overlapping data $X_{P1|P2}$ with embeddings $E$, we compute both the embedding loss between the encoder's output and extracted hidden embeddings, and the reconstruction loss between the decoder's output and original overlapped input. For the remaining data $\overline{X_{P1|P2}}$ without embeddings, where $\overline{X_{P1|P2}} = X_{P1} - X_{P1|P2}$, we only compute the reconstruction loss. The total embedding approximation loss is defined as:

$$\mathcal{L}_{app} = \begin{cases} \lambda \mathcal{L}_{emb}(F_{enc}(x), E) + (1-\lambda)\mathcal{L}_{rec}(F_{dec}(F_{enc}(x)), x), & \text{if } x \in X_{P1|P2} \\ \mathcal{L}_{rec}(F_{dec}(F_{enc}(x)), x), & \text{otherwise} \end{cases}, \tag{9}$$

where $\lambda \in [0, 1]$ is a hyperparameter to balance the importance between reconstruction and embedding learning. $\mathcal{L}_{emb}(\cdot)$ and $\mathcal{L}_{rec}(\cdot)$ are embedding loss and reconstruction loss, respectively. We set the Mean Squared Error (MSE) for both loss functions. After training, we utilize the trained encoder function $F_{enc}(X_{P1|A}) = \hat{E}$ to obtain embedding approximations $\hat{E}$ for overlapping data $X_{P1|A}$ from the first-hop party in conjunction with the active party.

### 3.3 Teacher and Student Models

**Teacher Model.** In our enhanced VFL system, instead of directly using $X_{P1|A}$ to train the local model, we utilize the embedding approximations $\hat{E}$ obtained above from $X_{P1|A}$. Both the local and global models share a three-hidden-layer neural network structure, with dropout used for regularization. The VFL global loss with cross-entropy is defined as:

$$\mathcal{L}_{vfl}(\hat{Y}, Y) = -\sum_{i=1}^{C} Y_i \log(\hat{Y}_i), \tag{10}$$

where $\hat{Y}$ denotes the predicted probability distribution, $Y$ is the one-hot encoded true label, and $i$ indexes the classes. The VFL system acts as the teacher model, generating soft labels $\tilde{Y}$ from $\hat{Y}$, which are used to supervise the student model.
**Student Model.** We train a student model with the soft labels $\tilde{Y}$, allowing the active party to conduct inference $\hat{Y}^*$ only with local non-overlapping data $X_A^*$. We implement the student model as a neural network with three hidden layers, where the loss function minimizes the difference between the soft labels and the local predictions, employing the Kullback-Leibler (KL) divergence [12]. The distillation loss is defined as:

$$\mathcal{L}_{dis}(S(X_A), \tilde{Y}) = D_{KL}(\tilde{Y}||S(X_A)) = \sum_i \tilde{Y}_i \log\left(\frac{\tilde{Y}_i}{S(X_A)_i}\right), \tag{11}$$

where $S(\cdot)$ is the student model.

**Algorithm.** We introduce the pseudo-algorithm of the main structure (first three modules) in Algorithm 1.

## 4 Experiments

We evaluate VFL-ASP against two baseline methods on three real-world datasets, testing diverse feature splits and sample overlaps. In addition, we assess the student model, analyze runtime efficiency, and perform an ablation study showing the effectiveness of our embedding extraction module.

### 4.1 Experimental Setup

We implement VFL-ASP using python 3.9 and pytorch 2.3.1, running on an NVIDIA GeForce RTX 4090 GPU with an Intel(R) Xeon w5-2455X @ 3.20 GHz CPU. Without loss of generality, we designate a single party to each of the second-hop and first-hop roles. Following a permutation analysis for all features to identify the importance score, we manually assign critical features to second-hop party and measure classification accuracy with a 95% confidence interval across 100 runs.

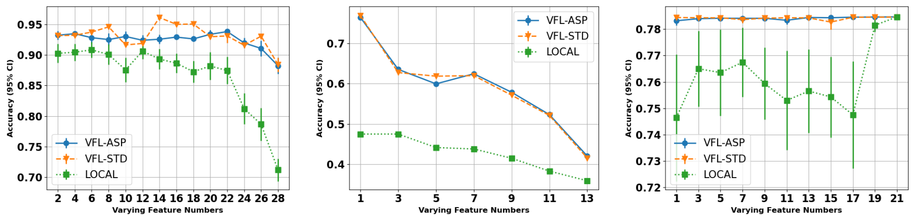
**Algorithm 1** VFL-ASP

**Input:** Overlapping data $X_{P2}, X_{P1|P2}, X_{P1|A}, X_A$
**Output:** VFL prediction $\hat{Y}$ (soft labels $\tilde{Y}$)
**Step 1: Embedding Extraction**
**for** each $X_i \in \{X_{P1|P2}, X_{P2}\}$ **do**
  Receive $P$ from trusted third party, encrypt: $X_i^\epsilon \leftarrow PX_i$
  Send $X_i^\epsilon$ to central server
**end for**
Compute SVD: $X^\epsilon = U^\epsilon \Sigma^\epsilon V^{\epsilon T}$
Extract embeddings: $E \leftarrow [P^T U^\epsilon]\Sigma^\epsilon$
**Step 2: Embedding Approximation**
Train autoencoder on $X_{P1}$, obtain $\mathcal{L}_{emb}$ and $\mathcal{L}_{rec}$
Minimize: $\mathcal{L}_{app} \leftarrow \lambda \mathcal{L}_{emb} + (1-\lambda)\mathcal{L}_{rec}$ **if** $x \in X_{P1|P2}$, else $\mathcal{L}_{rec}$
Approximate: $\hat{E} \leftarrow F_{enc}(X_{P1|A})$
**Step 3: VFL Model**
**for** each local party $\hat{E}, X_A$ **do**
  Train local $M_i$, send $H_i$ to global model $G$
  Receive gradients to update $\theta_i$
**end for**
Train global $G$, minimize $\mathcal{L}_{vfl}(\hat{Y}, Y)$, update $\phi$
Repeat until convergence
**return** $\hat{Y}$ (soft labels $\tilde{Y}$)

**Datasets.** We evaluate VFL-ASP on three real datasets:
Breast Cancer [26]: The Wisconsin Breast Cancer data contains features from digitized images of breast mass aspirates. The data has $\mathbb{R}^{569\times30}$ dimensions with binary diagnosis labels (malignancy, benign).
MIMIC-III [11]: The MIMIC-III database covers vital signs, lab tests, and medication from over 40k patients with 58k ICU stays. The data has $\mathbb{R}^{58976\times15}$ dimensions, and we use the length of stay (four classes) as the prediction label.
Credit [28]: The dataset describes customer default payments, including credit limit and payment history. The data has $\mathbb{R}^{30000\times23}$ dimensions, with a binary label indicating whether a customer defaults on their credit card payment.

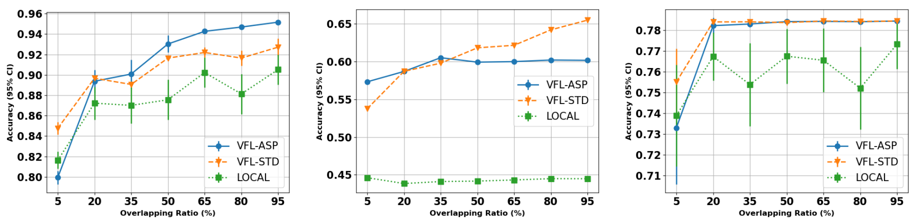
**Baselines.** We evaluate the classification accuracy and runtime performance of VFL-ASP against the following baselines:
VFL-STD [20]: The standard VFL model, where each first-hop passive and active party trains a local model using its private data. Intermediate results are then sent to the active party for global model training in each communication round.
LOCAL: The local model where the active party only leverages its own feature information to train the model and conduct inference.

We implement a consistent three-hidden-layer neural network structure for both local and global models across all methods, differing only in the input and output dimensions. This design ensures a fair comparison while balancing efficiency and performance. VFL-ASP is model-agnostic and can adapt to task-specific architectures. Our source code and data are publicly available [4].




**(a)** Breast Cancer **(b)** MIMIC-III **(c)** Credit

**Fig. 5.** Comparative accuracy vs. varying feature split.




**(a)** Breast Cancer **(b)** MIMIC-III **(c)** Credit

**Fig. 6.** Comparative accuracy vs. varying sample overlap.

## 4.2 Experimental Results

**Exp-1: Feature Split.** Figure 5 illustrates the comparative accuracy as the feature number split varies across parties. Specifically, the feature number is kept equal between the first-hop and active parties, while the number in second-hop increases incrementally. We observe that VFL-ASP achieves similar performance to VFL-STD across all datasets, but demonstrates greater overall stability on the Breast Cancer dataset, maintaining accuracy around 93%.

**Exp-2: Sample Overlap.** We measure the accuracy of VFL-ASP against the baselines for increasing sample overlap between the first-hop and active party, over total overlap with both the active and second-hop parties. We define this overlap ratio as $\alpha = \frac{X_{P1|A}}{X_{P1}}$, where $X_{P1} = X_{P1|P2} + X_{P1|A}$. Figure 6 shows that when $\alpha > 35\%$, VFL-ASP consistently outperforms the baselines on the Breast Cancer dataset as $\alpha$ increases. For the MIMIC-III dataset, VFL-ASP only outperforms VFL-STD in low-overlap scenarios, indicating that its approximation may struggle to capture MIMIC-III's complex feature dependencies when overlap is limited. Compared to VFL-STD, VFL-ASP achieves a comparable accuracy of about 78.5% over the Credit dataset.

Under the general condition of equal feature split with $\alpha = 50\%$ sample overlap, Table 2 shows that VFL-ASP consistently outperforms the baselines on the Breast Cancer and Credit datasets.

**Table 2.** Comparison of accuracy across models.

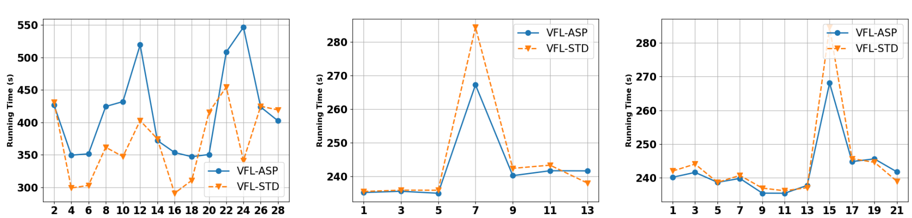
| Models | Breast Cancer | MIMIC-III | Credit |
|---|---|---|---|
| **VFL-ASP** | **93.027**±0.840 | 59.928±0.089 | **78.410**±0.028 |
| VFL-STD [20] | 91.640±0.243 | **61.829**±0.268 | 78.363±0.132 |
| LOCAL | 87.553±1.969 | 44.160±0.114 | 76.745±1.313 |

**Exp-3: Student Effectiveness.** We compare the student model, supervised via the distilled knowledge from the VFL teacher model, against the local model of the active party. Table 3 highlights that the VFL-ASP student model achieves higher accuracy on the Breast Cancer and MIMIC-III datasets. By leveraging critical features from the second-hop party during inference, it achieves an approximate 3% improvement on the Breast Cancer dataset. In the Credit dataset, VFL-ASP experiences an approximate 1% decrease, likely due to less distinct feature separation and utility between the first-hop and second-hop parties.

**Table 3.** Student vs. LOCAL model accuracy.

| Models | Breast Cancer | MIMIC-III | Credit |
|---|---|---|---|
| **Student** | **89.647**±2.056 | **44.869**±0.118 | 75.511±2.151 |
| LOCAL | 86.723±2.410 | 44.124±0.110 | **76.671**±1.321 |

**Exp-4: Runtime Analysis.** Figure 7 presents the comparative runtime (total training and inference time) of VFL-ASP versus VFL-STD with varying feature splits. On average, VFL-ASP incurs a 45.137$s$ overhead compared to VFL-STD on the Breast Cancer dataset, but is 2.697$s$ and 1.858$s$ faster on MIMIC-III and Credit datasets, respectively. The peaks in the graphs may result from certain feature splits creating more complex optimization scenarios, requiring more iterations for stability. The early stopping process enables VFL-ASP to converge faster.




**(a)** Breast Cancer **(b)** MIMIC-III **(c)** Credit

**Fig. 7.** Comparative runtime performance vs. varying feature split.

**Exp-5: Ablation Study.** To evaluate the effectiveness of the embedding extraction module, we conduct an ablation study using a three-hidden-layer neural network under four different input configurations, as outlined in Table 4. Embedding: Uses embeddings extracted from overlapping data between first-hop and second-hop passive parties, incorporating the extraction module. Second-hop and First-hop: Use raw overlapping data from their respective local parties, bypassing the extraction step. Centralized: Concatenates first and second-hop data to provide a reference for achievable performance, serving as an ideal upper bound. Note that this setting is infeasible in FL environments.

**Table 4.** Ablation study accuracy.

| Input Configurations | Breast Cancer | MIMIC-III | Credit |
|---|---|---|---|
| **Embedding:** $E$ | **93.400**±0.830 | **83.502**±0.112 | $\boxed{76.990}$±0.192 |
| Second-hop data: $X_{P2}$ | $\boxed{92.633}$±1.040 | 82.678±0.139 | 76.941±0.218 |
| First-hop data: $X_{P1\|P2}$ | 91.033±0.924 | 66.439±0.238 | 76.843±0.162 |
| Centralized data: $[X_{P2}, X_{P1\|P2}]$ | 92.500±1.057 | $\boxed{83.472}$±0.155 | **77.001**±0.168 |

By comparing these configurations, we isolate the impact of embedding extraction. Table 4 shows that embedding configuration achieves the highest accuracy on the Breast Cancer and MIMIC-III datasets, surpassing the centralized data, and is within 0.011% on the Credit dataset. This improvement reflects enhanced feature representation. Notably, among all datasets, the embeddings achieve up to 17.063% (first-hop) accuracy gains, with the greatest improvement observed on the larger MIMIC-III dataset.

## 5 Related Work

Recent studies have explored methods to relax the strict overlap assumption in VFL. VFedTrans [10] utilizes FedSVD [1] or VFedPCA [2] to extract latent representations of overlapping samples. It employs autoencoder-based methods [7,9] or GANs [6] to transfer knowledge and enrich the local data at the active party. However, VFedTrans only considers one passive and one active party at a time, making it inefficient for multiple passive parties. Similarly, VFL-infer [24] leverages knowledge distillation [8] with privileged feature information to supervise a local student model at the active party. However, both methods assume direct data overlap between passive and active parties, limiting the applicability where only a subset of passive parties shares sample alignment.

Semi-supervised VFL approaches, such as FedCVT [14], introduce techniques to enhance learning in VFL settings. FedCVT estimates missing representations, generates pseudo-labels, and performs cross-view training to improve model performance. However, it still requires a minimal amount of aligned data across parties. Federated transfer learning (FTL) [5, 19, 25] combines (vertical) federated learning and transfer learning [23] to facilitate cross-domain knowledge sharing while addressing privacy and security concerns. FTL projects features into a shared subspace for transfer but relies on direct data overlap between a single source-domain and a target-domain party.

Despite advances, none of these methods can be directly applied to the data configuration as second-hop overlapping scenarios.

## 6 Conclusion

We explore a VFL setting where second-hop passive parties do not directly overlap with the active party but share sample overlap with first-hop parties. This scenario may arise in practice due to privacy regulations and security policies. To address this, we introduce VFL-ASP, a novel framework designed to integrate features from second-hop parties into the VFL system. The framework extracts hidden embeddings to capture these features and uses knowledge distillation to conduct inference on the active local party. Our evaluation shows promising accuracy with diverse feature splits and sample overlaps. Through an ablation study, we demonstrate the effectiveness of the embedding extraction module. As the next step, we intend to study client selection strategies that consider both data quality and utility in the asynchronous FL.

## References

1. Chai, D., Wang, L., Zhang, J., Yang, L., Cai, S., Chen, K., Yang, Q.: Practical lossless federated singular vector decomposition over billion-scale data. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 46–55 (2022)
2. Cheung, Y., Jiang, J., Yu, F., Lou, J.: Vertical federated principal component analysis and its kernel extension on feature-wise distributed data. arXiv preprint arXiv:2203.01752 (2022)
3. Dang, T.K., Lan, X., Weng, J., Feng, M.: Federated learning for electronic health records. ACM Transactions on Intelligent Systems and Technology (TIST) (2022)
4. Dou, Z.: Vertical federated learning across second-hop parties (extended report) (2025), `https://github.com/VFL-ASP/VFL-ASP`
5. Feng, S., Li, B., Yu, H., Liu, Y., Yang, Q.: Semi-supervised federated heterogeneous transfer learning. Knowledge-Based Systems **252**, 109384 (2022)
6. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Communications of the ACM **63**(11), 139–144 (2020)
7. Higgins, I., Matthey, L., Pal, A., Burgess, C.P., Glorot, X., Botvinick, M.M., Mohamed, S., Lerchner, A.: Beta-vae: Learning basic visual concepts with a constrained variational framework. ICLR (Poster) **3** (2017)
8. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
9. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
10. Huang, C., Wang, L., Han, X.: Vertical federated knowledge transfer via representation distillation for healthcare collaboration networks. In: Proceedings of the ACM Web Conference 2023. pp. 4188–4199 (2023)

11. Johnson, A.E.W., Pollard, T.J., Shen, L., Lehman, L.W.H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., Mark, R.G.: Mimic-iii: A freely accessible critical care database. Scientific Data **3**(1), 1–9 (2016)
12. Joyce, J.M.: Kullback-leibler divergence. In: International Encyclopedia of Statistical Science, pp. 720–722. Springer (2011)
13. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R.: Advances and open problems in federated learning. Foundations and Trends® in Machine Learning **14**(1–2), 1–210 (2021)
14. Kang, Y., Liu, Y., Liang, X.: Fedcvt: Semi-supervised vertical federated learning with cross-view training. ACM Transactions on Intelligent Systems and Technology (TIST) **13**(4), 1–16 (2022)
15. Konečnỳ, J.: Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)
16. Konečnỳ, J., McMahan, B., Ramage, D.: Federated optimization: Distributed optimization beyond the datacenter. arXiv preprint arXiv:1511.03575 (2015)
17. Li, W., Xia, Q., Cheng, H., Xue, K., Xia, S.T.: Vertical semi-federated learning for efficient online advertising. arXiv preprint arXiv:2209.15635 (2022)
18. Lim, W.Y.B., Luong, N.C., Hoang, D.T., Jiao, Y., Liang, Y.C., Yang, Q., Niyato, D., Miao, C.: Federated learning in mobile edge networks: A comprehensive survey. IEEE Communications Surveys & Tutorials **22**(3), 2031–2063 (2020)
19. Liu, Y., Kang, Y., Xing, C., Chen, T., Yang, Q.: A secure federated transfer learning framework. IEEE Intelligent Systems **35**(4), 70–82 (2020)
20. Liu, Y., Kang, Y., Zou, T., Pu, Y., He, Y., Ye, X., Ouyang, Y., Zhang, Y.Q., Yang, Q.: Vertical federated learning: Concepts, advances, and challenges (2023)
21. Liu, Y., Zhang, X., Kang, Y., Li, L., Chen, T., Hong, M., Yang, Q.: Fedbcd: A communication-efficient collaborative learning framework for distributed features. IEEE Transactions on Signal Processing **70**, 4277–4290 (2022)
22. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics. pp. 1273–1282. PMLR (2017)
23. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering **22**(10), 1345–1359 (2009)
24. Ren, Z., Yang, L., Chen, K.: Improving availability of vertical federated learning: Relaxing inference on non-overlapping data. ACM Transactions on Intelligent Systems and Technology (TIST) **13**(4), 1–20 (2022)
25. Sharma, S., Xing, C., Liu, Y., Kang, Y.: Secure and efficient federated transfer learning. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 2569–2576. IEEE (2019)
26. Wolberg, W., Mangasarian, O., Street, N., Street, W.: Breast cancer wisconsin (diagnostic). UCI Machine Learning Repository (1995), DOI: https://doi.org/10.24432/C5DW2B
27. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019)
28. Yeh, I.C.: Default of credit card clients. UCI Machine Learning Repository (2016), DOI: https://doi.org/10.24432/C55S3H
29. Yin, X., Zhu, Y., Hu, J.: A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. ACM Computing Surveys (CSUR) **54**(6), 1–36 (2021)