A Coupled Design of Exploiting Record Similarity for Practical Vertical Federated Learning

Zhaomin Wu, Qinbin Li, Bingsheng He
National University of Singapore
{zhaomin,qinbin,hebs}@comp.nus.edu.sg

Abstract

Federated learning is a learning paradigm to enable collaborative learning across different parties without revealing raw data. Notably, *vertical federated learning* (VFL), where parties share the same set of samples but only hold partial features, has a wide range of real-world applications. However, most existing studies in VFL disregard the "record linkage" process. They design algorithms either assuming the data from different parties can be exactly linked or simply linking each record with its most similar neighboring record. These approaches may fail to capture the key features from other less similar records. Moreover, such improper linkage cannot be corrected by training since existing approaches provide no feedback on linkage during training. In this paper, we design a novel coupled training paradigm, FedSim, that integrates one-to-many linkage into the training process. Besides enabling VFL in many real-world applications with fuzzy identifiers, FedSim also achieves better performance in traditional VFL tasks. Moreover, we theoretically analyze the additional privacy risk incurred by sharing similarities. Our experiments on eight datasets with various similarity metrics show that FedSim outperforms other state-of-the-art baselines. The codes of FedSim are available at `https://github.com/Xtra-Computing/FedSim`.

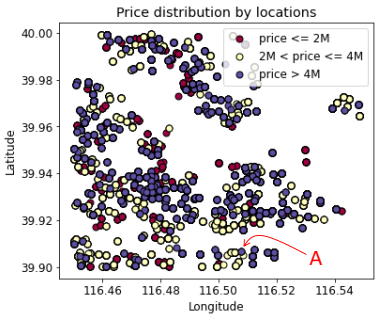
1 Introduction

Federated learning is a collaborative learning framework to train a model from distributed datasets with privacy guarantees. A commonly existing and widely studied scenario of federated learning is *vertical federated learning* [32, 62] (VFL), where multiple parties sharing the same set of samples have different sets of features. We focus on the setting where only one party holds the labels like most of the studies [33, 39, 60]. The party holding labels is named *primary party*; the parties without labels are named *secondary parties*. In VFL, the features that exist on multiple parties are called *common features*. The vector of common features in a data record is called the *identifier* of the record.

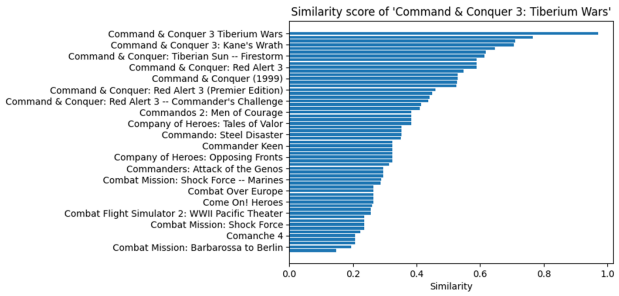
Existing studies [22, 27, 39, 40] formulate VFL as two separated processes: linkage and training. In the linkage process, the datasets on different parties are linked according to the identifiers. In the training process, these distributed but linked data records are trained by VFL algorithms. Specifically, in the linkage process of existing studies, each data record is linked to a data record with the exactly matched or the most similar identifier (i.e., *one-to-one* linkage). Nonetheless, according to our study on all the completed projects in German Record Linkage Center [4] (GRLC), only 27.3% of the record-linkage applications are linked on exact identifiers. The other projects rely on fuzzy identifiers such as addresses, in which one-to-one linkage can seriously impair the accuracy of the VFL model for the following two reasons.

First, only linking the records with top similarity (i.e., *one-to-one* linkage) does not necessarily capture the key features, which can be demonstrated by two real-world applications. 1) Considering the VFL for price prediction between a real estate company and a house leasing company linked by

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



(i) Housing prices by geolocations in Beijing



(ii) Games with similar titles to "Command & Conquer 3: Tiberium Wars"

Figure 1: Examples of real-world record linkage (`house` and `game` dataset in the experiments)

GPS locations of houses, as shown in Figure 1(i), the closest house to house A (purple) may not be in the same price level as A. 2) Considering the VFL between steam games and IGN games linked by the game titles, as shown in Figure 1(ii), linking not only the exactly matched game but also other games in the same series intuitively benefits game recommendation tasks. Even in applications where identifiers can be exactly matched (e.g., identifiers are ID), if some features are missing or biased, one-to-one linkage prevents the training process from enhancing these features from other similar records. These applications commonly exist in practice according to our investigation. An opposite extreme case is to link each record with all the records in another party (*one-to-all* linkage), which keeps all the information but is too expensive for both linkage and training. Therefore, a *one-to-many* linkage approach is needed as a balance between efficiency and performance.

Second, separating the linkage from the training also harms the performance of VFL. In existing VFL approaches, since the linkage process cannot obtain any feedback from the training process, the linkage is conducted with the goal of finding *true-matched* pairs of records instead of finding the pairs that reduce the training loss. Hence, an integrated VFL framework that conducts one-to-many linkage under the guide of training is desired.

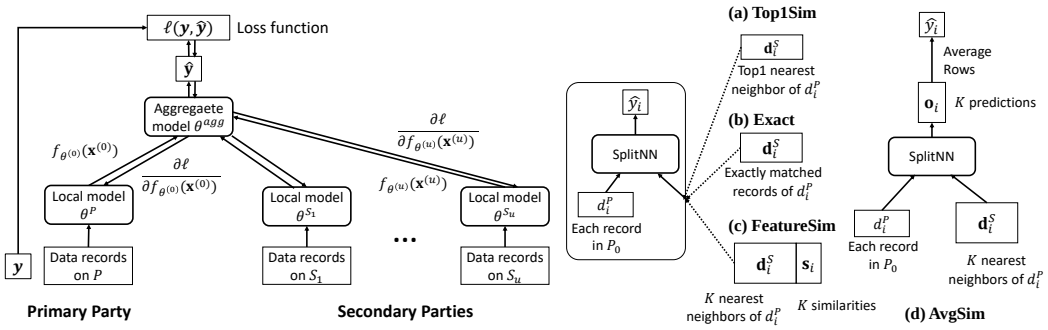
To address these two drawbacks, we link each data record to the records with top-K similar identifiers in another party and design a coupled framework of linkage and training. Our main challenge is to effectively exploit these linked pairs and their similarities to boost the performance of VFL. To tackle this challenge, we propose a similarity-based coupled VFL framework `FedSim` on the top of SplitNN [56] which is a VFL algorithm for neural networks. In FedSim, *similarity* is a dominant feature that determines the order and the weight of each pair of linked records. The weights (i.e., impact) of similarities are also adjusted in each training iteration. After the training, each similarity is mapped to a weight based on its contribution to reducing the loss.

Furthermore, to address the additional data privacy issue incurred by FedSim, we first add Gaussian noise to the similarities and analyze the privacy of FedSim under differential privacy [15]. Our analysis suggests that differential privacy that provides rigorous guarantees for every individual record regardless of the type of attack is impractical for FedSim. As an alternative, we propose an intuitive greedy attack to infer identifiers from similarities, followed by a theoretical bound of the probability of its success.

Our main contributions can be summarized below. 1) We propose a novel asymmetric training paradigm, named FedSim, and a training-free metric to estimate the improvement of FedSim on baseline approaches; 2) we analyze the privacy of FedSim under differential privacy; 3) we propose a greedy attack on FedSim and the corresponding defense method that can theoretically bound the success rate of this attack; 4) we conduct extensive experiments on three synthetic datasets and five real-world datasets, which indicates that FedSim outperforms state-of-the-art baselines.

2 Preliminaries

Privacy-Preserving Record Linkage. *Privacy-preserving record linkage* (PPRL) [55] aims to link the data records from two parties that refer to the same sample without revealing real identifiers. Most



(i) Forward and back-propagation of SplitNN

(ii) Structure of baseline models

Figure 2: Structure of SplitNN and baseline VFL models

PPRL methods [28, 53, 55] consist of three main steps: blocking, comparison, and classification. First, in the blocking step, data records that are unlikely to be linked are pruned to reduce the number of comparisons. Then, in the comparison step, a similarity between identifiers is computed for each candidate pair of data records. Finally, in classification, each candidate pair is classified as "matches" or "non-matches", which is usually done by a manually set threshold. Notably, although blocking ensures sample scalability, party scalability remains a challenge in PPRL. Therefore, we focus on the linkage and training process of two parties and provide an extension to multiple parties in this paper.

Although the training of FedSim does not rely on any specific PPRL framework, our privacy analysis is based on a state-of-the-art PPRL framework FEDERAL [29]. Coordinated by an honest-but-curious server, FEDERAL calculates similarities by comparing Bloom filters [55] generated from identifiers. It is theoretically guaranteed that all the identifiers generate Bloom filters containing similar numbers of ones; thus, attackers are hard to distinguish identifiers based on Bloom filters. More detailed background of FEDERAL is introduced in Appendix I.

Vertical Federated Learning. In this paragraph, we present a formal definition of VFL between two parties. Suppose two parties P and S want to cooperate with each other to train a machine learning model. P is the primary party holding m samples and labels $\{\mathbf{x}^P, \mathbf{y}\} \triangleq \{x_i^P, y_i\}_{i=1}^m$, S is the secondary party holding n samples $\mathbf{x}^S \triangleq \{x_i^S\}_{i=1}^n$ which can also benefit the machine learning task. In order to perform linkage, we assume there are some common features between $\{x_i^P\}_{i=1}^m$ and $\{x_i^S\}_{i=1}^n$, i.e., $\{x_i^P\}_{i=1}^m = \{d_i^P, k_i^P\}_{i=1}^m, \{x_i^S\}_{i=1}^n = \{d_i^S, k_i^S\}_{i=1}^n$, where k_i^P, k_i^S are common features used for linkage and d_i^P, d_i^S are remaining features used for training with dimension l_P, l_S. We denote $\mathbf{d}^P \triangleq \{d_i^P\}_{i=1}^m$, $\mathbf{d}^S \triangleq \{d_i^S\}_{i=1}^n$ for simplicity. Our goal is to enable party P to exploit $\mathbf{x}^S$ to train a model that minimizes the global loss. Formally, we aim to optimize the following formula:

$$\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} L(f(\theta; x_i^P, \mathbf{x}^S);\ y_i) + \lambda \Omega(\theta)$$

where $L(\cdot)$ is the loss function, $f(\cdot)$ is the VFL model, and $\lambda\Omega(\theta)$ is the regularization term.

SplitNN. Most vertical federated learning algorithms only support simple models like logistic regression [25] which are ineligible to handle many real-world applications. Therefore, we adopt SplitNN [56] which is a popular and state-of-the-art VFL algorithm that supports neural networks. The main idea of SplitNN is to split a model into multiple parties and conduct training by transferring gradients and intermediate outputs across parties. As shown in Figure 2(i), a global model is split into an aggregation model θ^{agg} and H local models $\theta^P, \{\theta^{S_u} | u \in [1, H-1]\}$, where H is the number of parties. In each iteration, the outputs of local models are calculated by forward propagation and sent to the primary party P_0 which holds the labels. After concatenating the outputs of local models, P continues forward propagation to derive the final prediction $\hat{y}$ and the loss ℓ. Then, P performs back-propagation until the input of the aggregation model. The gradients w.r.t. the outputs of each local model are calculated and sent to the corresponding secondary party. Finally, all H parties finish back-propagation with the gradients.

3 Baseline Approaches

By extensively reviewing the existing VFL approaches, we find that all the existing approaches only use exactly matched (Exact) or most similar (Top1Sim) pairs of data records in the training. We denote these approaches that separate the linkage and training and as *separated approaches*. According to our detailed study in Appendix D, only 27.3% of the record-linkage applications in GRLC based on exact identifiers are suitable for these approaches. To design a *coupled approach* that effectively exploits linkage information, we analyze the main drawback of existing approaches and some baseline coupled frameworks.

Exact [7, 18, 26, 35, 36, 59, 60]/Top1Sim [22, 27, 39, 40]. (Figure 2(ii)(a,b)) These baseline separated approaches that only link exactly matched or most similar pairs neglect information of less similar but useful pairs, resulting in a poor performance of VFL model in many real applications such as those in GRLC.

AvgSim. (Figure 2(ii)(d)) We propose AvgSim as a baseline coupled approach that considers multiple pairs of records. Specifically, each data record in x_i^P party P is linked with its K most similar data records in party S. The prediction of x_i^P is an average of prediction when linking x_i^P with its K nearest x_j^S. Though considering more pairs, AvgSim overemphasizes the pairs with medium similarities, leading to redundant noise added to the model. Unaware of the similarity of each pair, the model is unable to filter out this redundant noise.

FeatureSim. (Figure 2(ii)(c)) We propose FeatureSim as a coupled baseline approach that adopts similarities to the training. Each x_i^P in party P is linked with its K most similar data records in party S, and the similarity between each pair is appended to the record. Nonetheless, similarity, which contains critical linkage information, is treated equally to other features. This drawback limits the ability of VFL models to extract information from similarities, thus affecting the overall performance. Such an impact can be significant according to our experiments.

Based on the above analyses, we clearly observe that the record linkage should be coupled with the design of VFL by taking advantage of record similarity not only during the linkage procedure but also during the training procedure. Meanwhile, according to the analysis on AvgSim, more advanced models are needed to merge the outputs of linked pairs with K largest similarities. These analyses motivate our design of FedSim as a coupled framework of linkage and training.

4 Our Approach: FedSim

Our approach has two components: soft linkage and similarity-based VFL. In soft linkage, after finding top-K similar pairs by existing PPRL methods, the server preprocesses the similarities by normalization and Gaussian noise perturbation. The scale of noise can be determined by a constant bound τ of the attacker's success rate, which is further analyzed in Section 5. Then, the aligning information and aligned similarities are sent to party P or S which will align the data records accordingly. In similarity-based VFL, we design a model (as shown in Figure 3) by adding additional components around SplitNN to effectively exploit similarities. Finally, this model, taking the aligned data records and aligned similarities as input, is trained by back-propagation like SplitNN. Although we use SplitNN as an example, the idea of soft linkage and similarity-based learning can be extended to other federated learning algorithms.

4.1 Soft Linkage

Soft linkage, following traditional PPRL in "blocking" and "comparison" steps, directly outputs the similarities with normalization and noise addition without performing the "classification" step. Same as many existing PPRL approaches [28–30, 54], we assume there exists an honest-but-curious server coordinating the linkage process. Specifically, soft linkage, taking $\mathbf{k}^P$ and $\mathbf{k}^S$ as input, outputs the alignment information (i.e., indices that indicate the order of records) and similarities. First, each sample x_i^P in party P is linked with samples containing K most similar identifiers in party S. The similarities between identifiers of these linked pairs are calculated by a PPRL protocol. To fully utilize $\mathbf{x}^S$ for each sample x_i^P, K should be large enough to ensure that all the pairs which may benefit the model are included. Notably, setting a large K hardly leads to overfitting according to our experiments in Appendix F.2. Second, the server calculates the raw similarities ρ_{ij} between these

linked pairs; the raw similarity ρ_{ij} is defined as normalized negative distance. Formally,

$$\rho_{ij} = \frac{-\operatorname{dist}(k_i^P, k_j^S) - \mu_0}{\sigma_0}. \tag{1}$$

where μ_0 and σ_0 are the mean and standard variance of all negative distances $-\operatorname{dist}(k_i^P, k_j^S)$. To prevent the attacker from guessing the vectors k_i^P or k_j^S from similarities (further discussed in Section 5), we add Gaussian noise of scale σ to each ρ_{ij}. Formally, for $\forall i \in [1, m], \forall j \in [1, K]$

$$s_{ij} = \rho_{ij} + N(0, \sigma^2). \tag{2}$$

For simplicity, we denote $\mathbf{s}_i \triangleq \{s_{ij}\}_{j=0}^{K}$ and $\mathbf{s} \triangleq \{\mathbf{s}_i\}_{i=0}^{m}$.

After the similarities are calculated, the server directly sends the similarities $\mathbf{s}$ to party P and sends the aligning information (i.e., indices that indicate orders) to both parties. Finally, parties P and S align their samples or similarities according to the aligning information to ensure that each x_i^P, $\mathbf{x}_i^S$ and $\mathbf{s}_i$ refer to the same sample.

4.2 Similarity-Based VFL

The training process is summarized in Algorithm 1 and the model structure is shown in Figure 3. As discussed in Section 7, FedSim is designed on top of SplitNN [56], which makes preliminary predictions $\mathbf{o}_i$ (with l_m dimensions) for each d_i^P and its K neighbors $\mathbf{d}_i^S$ (lines 4-7). Specifically, we add three *gates* (weight gate, merge gate, sort gate) around SplitNN and train the whole model similarly to SplitNN. The main function of these gates is to effectively exploit similarities $\mathbf{s}_i$ to merge the preliminary outputs $\mathbf{o}_i$ into the final prediction $\hat{y}_i$.

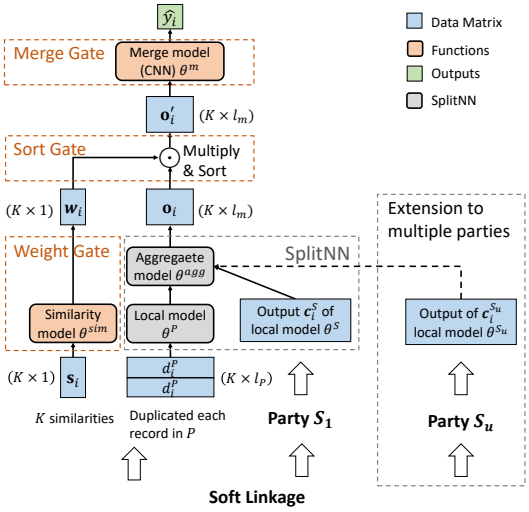


Figure 3: Model structure of FedSim

Weight Gate. One straightforward idea is that more similar pairs of samples contribute more to the performance. Thus, the rows in $\mathbf{o}_i$ with higher similarities should be granted a larger weight. Directly multiplying the similarities to $\mathbf{o}_i$ is inappropriate since the values of similarities do not directly represent the importance of records. Therefore, in weight gate, we use *similarity model*, a simple neural network with one-dimensional input and one-dimensional output, to non-linearly map the similarities to weights $\mathbf{w}_i$ which indicates the importance of the record (line 8). Then, the weighted outputs are calculated as a matrix multiplication (line 9).

Merge Gate. The merge gate contains a merge model to aggregate the information in $\mathbf{o}'_i$ (line 11). For tasks that only require a linear aggregation, setting the merge model as an "average over rows" of $\mathbf{o}'_i$ is sufficient to obtain a promising result. However, for tasks that require a non-linear aggregation, a neural network is required to be deployed as a merge model. In this case, although the merge model can be implemented by a multi-layer perceptron (MLP) which takes flattened $\mathbf{o}'_i$ as input, such an approach usually leads to over-fitting due to a large number of parameters. Meanwhile, the flatten operation causes information loss because the merge model is unaware of which K features in the flattened $\mathbf{o}'_i$ correspond to the same output feature in $\mathbf{o}'_i$. Considering these two factors, we set the merge model as a 2D convolutional neural network (CNN) with kernel size $k_{conv} \times 1$. $\mathbf{o}'_i$, containing K output vectors with l_m dimensions, is regarded as a 2D input of size $K \times l_m$ for the CNN merge gate. Then, the merge gate effectively merges these output vectors with close similarities.

Sort Gate. The sort gate is an optional but sometimes crucial module depending on the property of the chosen merge model. For merge models that are insensitive to the order of $\mathbf{o}'_i$ (e.g., averaging over rows), sorting is not needed because the order does not affect the output of the merge model. However, for merge models that are sensitive to the order of $\mathbf{o}'_i$ (e.g., neural networks), inconsistent

order of features can incur irregular sharp gradients which makes the merge model hard to converge. Thus, $\mathbf{o}'_i$ should be sorted by similarities (line 10) to stabilize the updates on the merge model. Also, grouping the pairs with close similarities together helps the merge gate effectively aggregate the information.

Algorithm 1: Training Process of FedSim

Input : Aligned datasets and labels $\mathbf{d}^P$, $\mathbf{d}^S$, $\mathbf{y}$; similarities $\mathbf{s}$; number of similar samples K; number of epochs T; number of samples m in party P;
Output : SplitNN parameter $\theta_t^P, \theta_t^S, \theta^{agg}$; similarity model parameter θ_t^{sim}; merge model parameter θ_t^m

1 Initialize $\theta_0^P, \theta_0^S, \theta_0^{sim}, \theta_0^m, \theta_0^{agg}$; `// FP: forward propagation; BP: back-propagation`
2 **for** $t \leftarrow 0$ **to** T **do**
3 **for** $i \leftarrow 0$ **to** m **do**
4 **Party S** loads i-th batch $\mathbf{d}_i^S$ from $\mathbf{d}^S$;
5 calculates $\mathbf{c}_i^S = f(\theta_t^S; \mathbf{d}_i^S)$ and sends $\mathbf{c}_i^S$ to party P; `// Local model FP`
6 **Party P** loads i-th sample d_i^P from $\mathbf{d}^P$;
7 receives $\mathbf{c}_i^S$ from party S and calculates $\mathbf{o}_i = f(\theta_t^P; \mathbf{c}_i^S, d_i^P)$; `// SplitNN FP`
8 loads similarities $\mathbf{s}_i$ from $\mathbf{s}$ and calculates $\mathbf{w}_i = f(\theta_t^{sim}; \mathbf{s}_i)$; `// Similarity model FP`
9 calculates weighted outputs $\mathbf{o}'_i = \mathrm{diag}(\mathbf{w}_i)\mathbf{o}_i$;
10 sorts the rows of $\mathbf{o}'_i$ by $\mathbf{s}_i$ (or $\mathbf{w}_i$); `// Sort gate`
11 calculates $\hat{y}_i = f(\theta_t^m; \mathbf{o}'_i)$ with sorted $\mathbf{o}'_i$; `// Merge model FP`
12 calculates gradients $\mathbf{g}_t^P = \nabla_{\theta_t^P} L(\hat{y}_i, y_i)$, $\mathbf{g}_t^{sim} = \nabla_{\theta_t^{sim}} L(\hat{y}_i, y_i)$, $\mathbf{g}_t^m = \nabla_{\theta_t^m} L(\hat{y}_i, y_i)$, $\mathbf{g}_t^c = \nabla_{\mathbf{c}_i^S} L(\hat{y}_i, y_i)$ and sends $\mathbf{g}_t^c$ to party S; `// Merge and similarity model BP`
13 updates parameters $\theta_{t+1}^{agg} = \theta_t^{agg} - \eta_t \theta_t^{agg}, \theta_{t+1}^P = \theta_t^P - \eta_t \mathbf{g}_t^P, \theta_{t+1}^{sim} = \theta_t^{sim} - \eta_t \mathbf{g}_t^{sim}, \theta_{t+1}^m = \theta_t^m - \eta_t \mathbf{g}_t^m$;
14 **Party S** receives $\mathbf{g}_t^c$ from party P and continues calculating gradients $\mathbf{g}_t^S = \nabla_{\theta_t^S} \mathbf{g}_t^c$;
15 updates parameters $\theta_{t+1}^S = \theta_t^S - \eta_t \mathbf{g}_t^S$; `// Local model BP`

The whole model with SplitNN and three gates performs back-propagation by transferring gradients like SplitNN (lines 12, 14). After gradients are calculated, all the parameters are updated by gradient descent (lines 13, 15). According to the experiment in Appendix F.4, FedSim incurs longer but acceptable training time compared to Exact and Top1Sim. During the inference, for each data record in the primary party, K most similar data records in secondary parties are linked. These linked data records are fed into the FedSim model to derive the final prediction. More insights into the weight gate and merge gate are elaborated by visualization in Appendix F.5.

4.3 Improvement Estimation

Similar to many existing studies [7, 22, 39, 60] in VFL, we mainly focus on the two-party setting which has many real-world applications (e.g., bank and fintech company [60]). Meanwhile, we also support an extension to the multiple-party setting as elaborated in Appendix C.

In this subsection, we propose a data-linkage-based metric to estimate the improvement of FedSim over baselines (i.e., AvgSim, Top1Sim, Exact) without training. Calculating all the similarities between x_i^P and every $\{x_j^S\}_{j=1}^n$, we can plot the sorted similarities of each pair as a curve. For example, Figure 4 displays the curve of AvgSim with top-K neighboring records. The performance of these baselines is impeded by two main factors: 1) *lost information* - some pairs of samples with small similarities are neglected. For example, in Figure 4, the samples above the threshold are neglected; 2) *redundant information* - some pairs of samples with smaller similarities are treated equally to those pairs with large similarities. For example, in Figure 4, some samples below the threshold with medium similarities are overestimated. To jointly estimate these two factors, we intuitively assume the information contained

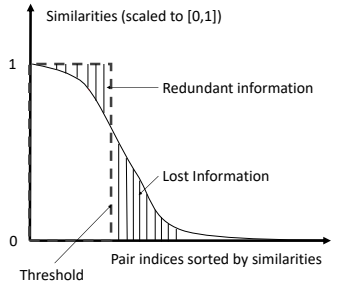


Figure 4: Estimated improvement of FedSim on AvgSim with top-K neighbors

in each pair is proportional to its similarity; then, the lost information and the redundant information can be estimated by the similarities. We formally define the metric as follows.

Definition 1. *Given a data linkage between $\mathbf{x}^P \triangleq \{x_i^P\}_{i=1}^m$ and $\mathbf{x}^S \triangleq \{x_j^S\}_{j=1}^n$, for a data record x_i^P in P, a VFL algorithm $\mathcal{F}$ divide the indices of n records in S into matches and non-matches, denoted as $S_{matches}$ and $S_{non\text{-}matches}$, respectively. Denote s_{ij} as the scaled similarity between x_i^P and x_j^S, the improvement of FedSim on $\mathcal{F}$ for x_i^P is defined as $\Delta_i(\mathcal{F}) \triangleq \sum_{j \in S_{matches}} (1 - s_{ij}) + \sum_{j \in S_{non\text{-}matches}} s_{ij}$; the overall improvement of FedSim on $\mathcal{F}$ is defined as $\Delta(\mathcal{F}) = \frac{1}{m} \sum_{i=1}^m \Delta_i(\mathcal{F})$.*

As shown in Figure 5(ii), this metric can effectively estimate the improvement of FedSim compared with the other baselines in the experiments.

5 Privacy

The shared information in FedSim includes:

(1) similarities $\mathbf{s}$ shared to party P;

(2) intermediate results in SplitNN shared to party P;

(3) intermediate results in PPRL shared to the server.

The intermediate results in SplitNN and PPRL are respectively studied in [57] and [29] (see Section 2 for details), both of which are orthogonal to this paper. Therefore, we mainly study the privacy risk caused by similarities $\mathbf{s}$.

5.1 Differential Privacy

According to Equation 1 and 2, each similarity s_{ij} is calculated from k_i^P and k_j^S. We formally define this procedure as $\mathcal{G}$ below.

Procedure $\mathcal{G}$: Take $\mathbf{k}^S$ as the input. Each $k_i^P \in \mathbf{k}^P$ is linked to multiple $k_j^S \in \mathbf{k}^S$; the similarities between k_i^P and k_j^S are calculated by $s_{ij} = -\frac{\|k_i^P - k_j^S\|_2 + \mu_0}{\sigma_0} + N(0, \sigma^2)$, where $N(0, \sigma^2)$ refers to Gaussian distribution with variance σ^2. Output the similarities $\mathbf{s}$.

Theorem 1. *Suppose $\varepsilon > 0$ and $0 < \delta < 1/2 - e^{-3\varepsilon}/\sqrt{2\pi\varepsilon}$. Suppose the size of $\mathbf{k}^P$ is $n \times \beta$. If procedure $\mathcal{G}$ is (ε, δ)-DP, then $\sigma \geq \Delta_{\mathcal{G}}/\sqrt{2\varepsilon}$, where $\Delta_{\mathcal{G}} = n \cdot \max\left\{\left|\frac{1+\mu_0}{\sigma_0}\right|, \left|\frac{-1+\mu_0}{\sigma_0}\right|\right\}$.*

The proof of Theorem 1 is included in Appendix A.1. As observed from Theorem 1, the noise scale σ increases linearly by n. The large scale of noise derived from differential privacy would seriously affect the performance of FedSim. Hence, we further analyze the privacy risk of FedSim against specific attacks.

5.2 Privacy Against Greedy Attacks

In this subsection, our analysis focuses on a greedy attacker who first predicts the most likely distance from each s_{ij} and then predicts the most likely Bloom filter from the predicted distance. Assuming the attacker already knows the scaling parameters μ_0, σ_0, we formulate the attack method as follows.

Attack Method. To obtain $\hat{k}_j^S$ as a prediction of k_j^S, the attacker 1) predicts a set of normalized negative distances $\hat{\rho}_{ij}$ $(i \in Q)$ from s_{ij} $(i \in Q)$, respectively, by maximum a posteriori (MAP) estimation with Gaussian prior $N(0, 1)$, i.e., $\hat{\rho}_{ij} = \arg\max_{\rho_{ij}} p(\rho_{ij} | s_{ij})$; 2) calculates each distance $\hat{u}_{ij}$ by scaling back $\hat{\rho}_{ij}$ with parameters μ_0, σ_0, i.e., $\hat{u}_{ij} = -\sigma_0 \hat{\rho}_{ij} - \mu_0$ $(i \in Q)$; 3) uniformly guesses $\hat{k}_j^S$ from all possible values satisfying $\forall i \in Q, \mathrm{dist}(k_i^P, \hat{k}_j^S) = \hat{u}_{ij}$ which have the same probability of being the real k_j^S.

Besides the greedy attack, advanced attackers may predict through the probability distribution of distances rather than predict through the most likely distance. Some attackers may even know some side information like the prior distribution of k_j^S and employ this side information to launch attacks. These advanced attacks are further discussed in Appendix B.

Although not satisfying differential privacy, we prove that the success probability of greedy attacks is always bounded by a small constant related to σ_0 and σ regardless of the choice of Q if an attacker follows the attack method (Theorem 2).

Theorem 2. *Given a finite set of perturbed similarities s_{ij} $(i \in Q)$ between $|Q|$ bloom-filters k_i^P $(i \in Q)$ in party P and one Bloom filter k_j^S in party S, if an attacker knows the scaling parameters μ_0, σ_0 and follows the procedure of the attack method, the probability of the attacker's predicted Bloom filter $\hat{k}_j^S$ equaling the real Bloom filter k_j^S is bounded by a constant τ. Formally,*

$$\Pr\left[\hat{k}_j^S = k_j^S \middle| \{s_{ij} | i \in Q\}, \{k_i^P | i \in Q\}, \mu_0, \sigma_0, \mathcal{A}\right] \leq \tau$$

where constant $\tau = \operatorname{erf}\left(\frac{\sqrt{\sigma^2+1}}{2\sqrt{2}\sigma\sigma_0}\right)$; $\operatorname{erf}(\cdot)$ is the error function, i.e., $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$; event $\mathcal{A}$: attackers follow the given attack method.

The proof of Theorem 2 is included in Appendix A. From Theorem 2, we find two factors that affect the attacker's success rate: 1) noise added to the similarity (σ); 2) standard variance of Bloom filters (σ_0). Among these factors, σ_0 determines the lower bound of the success rate because $\sqrt{\sigma^2+1}/(2\sqrt{2}\sigma\sigma_0) < 1/(2\sqrt{2}\sigma_0)$, and σ determines how close success rates FedSim can guarantee compared to the lower bound. When σ is large enough, increasing σ helps little with reducing the success rate. Therefore, to ensure the good privacy of FedSim, we should first guarantee a large enough variance among the Bloom filters and then add a moderate noise to the similarities. Taking *house* dataset (see Section 6.1) as an example where $\mu_0 = -46237.78, \sigma_0 = 21178.86$. Letting $\delta = 10^{-5}, \sigma = 0.4$, the differential privacy parameter $\varepsilon = 2.96 \times 10^9$ implies that there is almost no differential privacy guarantee at all. Nonetheless, the attacking success rate $\tau = 1.94 \times 10^{-5}$ suggests that the privacy risk from certain attacks is very low. Considering the 19479 samples in the training set of party S, only 0.378 (smaller than one) Bloom filters are expected to be disclosed.

6 Experiment

6.1 Experimental Setup

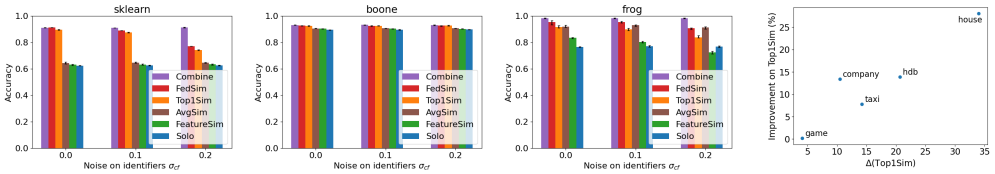
Dataset. We evaluate FedSim on three synthetic datasets (`sklearn` [10], `frog` [13], `boone`, [14]) and five real-world datasets (`house` [1, 44], `taxi` [8, 52], `hdb` [23, 45], `game` [11, 24], and `company` [6, 12]). The details (e.g. dimension of identifiers) of these datasets are summarized in Appendix E. For each real-world dataset, we collect two public datasets from different real-world parties and conduct VFL on both public datasets. For each synthetic dataset, we first create a global dataset by generating with sklearn API [10] (`sklearn`) or collecting from public (`frog`, `boone`). Then, we randomly select some features as common features and randomly divide the remaining features equally to both parties. Common features are not used in training for all methods except Combine. To simulate the real-world applications, for synthetic datasets, we also add different scales σ_{cf} of Gaussian noise to the common features. Specifically, for each identifier $\mathbf{v}_i$, the perturbed identifier $\mathbf{v}_i' = \mathbf{v}_i + N(\sigma_{cf}^2\mathbf{I})$ will be used for linkage. For datasets with numeric identifiers (`house`, `taxi`, `hdb`, `syn boone`, `frog`), Euclidean distance is adopted to calculate similarities. For `game` dataset, Levenshtein distance is adopted to calculate similarities. For `company` dataset, we first generate Bloom filters from strings following [29], then calculate similarities based on Hamming distances.

Training. Similarity model is a multi-layer perceptron (MLP) with one hidden layer. The merge model contains a 2D convolutional layer with $k_{conv} \times 1$ kernel followed by a dropout layer and an MLP with one hidden layer. Both the local model and aggregate model in SplitNN are MLPs with one hidden layer. We adopt LAMB [63], the state-of-the-art large-batch optimizer, to train all the models. Each dataset is split into training, validation, and test sets by 7:1:2. We run each algorithm five times and report the mean and standard variance (range is reported instead in figures) of performance on the test set. We present root mean square error (RMSE) or R-squared value (R^2) for regression tasks and accuracy for classification tasks. The choices of hyperparameters are introduced in Appendix E.

Baselines. We compare FedSim with nine baselines in our experiments. Besides the four baselines (Exact, Top1Sim, AvgSim, FeatureSim) introduced in Section 3, the remaining baselines include: 1) Solo: only dataset $\mathbf{d}^P$ is trained; 2) Combine: $[\mathbf{d}^P, \mathbf{d}^S]$ is trained (only applicable for synthetic datasets); 3) FedSim-MLP: the CNN in FedSim is changed to an MLP with a similar number of parameters. 4) FedSim w/o sort: FedSim without sorting gate. 5) FedSim w/o weight: FedSim

Table 1: Performance on real-world datasets

Algorithms	house (numeric) $\Delta = 34.05$	bike (numeric) $\Delta = 14.26$	hdb (numeric) $\Delta = 20.69$	game (string) $\Delta = 4.14$	company (string) $\Delta = 10.50$
Solo	58.31±0.28	272.83±1.50	29.75±0.15	85.27±0.29%	42.67±0.66
Exact	-	-	-	89.25±0.12%	44.44±1.95
Top1Sim	58.54±0.35	256.19±1.39	31.56±0.21	92.71±0.08%	42.84±0.77
FeatureSim	66.39±0.15	273.29±0.37	37.39±0.29	91.13±0.23%	39.24±1.80
AvgSim	51.92±0.65	239.85±0.40	34.12±0.19	90.84±0.14%	38.19±0.91
FedSim (w/o Weight)	42.82±0.20	236.79±0.29	27.18±0.08	92.79±0.13%	41.00±1.19
FedSim (w/o Sort)	52.14±0.58	238.30±0.81	36.35±0.42	92.79±0.10%	38.28±1.56
FedSim (w/o CNN)	42.62±0.20	235.97±0.42	27.76±0.13	92.50±0.12%	39.63±1.31
FedSim	**42.12±0.23**	**235.67±0.27**	**27.13±0.06**	**92.88±0.11%**	**37.08±0.61**



(i) Performance on synthetic datasets

(ii) Δ(Top1Sim) vs. perf.

Figure 5: Performance on synthetic datasets and the effectiveness of Δ(Top1Sim)

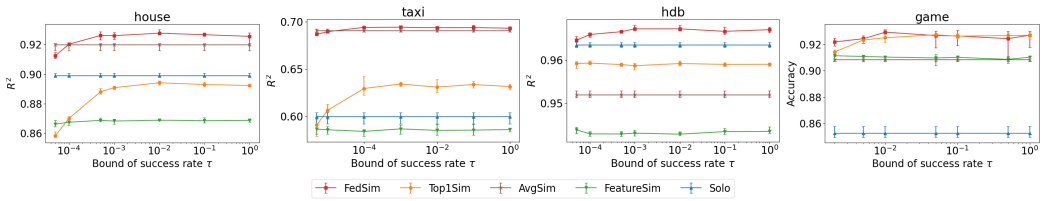


Figure 6: Performance with different scale of noise on similarities

without weight gate (similarities are directly regarded as weights). Notably, `Exact` is only evaluated on `game` and `company` because no exactly matched identifiers can be found on other datasets.

6.2 Performance

We evaluate the performance of FedSim on three synthetic datasets under different σ_{cf} and five real-world datasets. The results of synthetic datasets are presented in Figure 5(i), from which two observations can be made. First, FedSim consistently has better or close performance compared to all the baselines. Second, FedSim is more robust to the noise on the identifiers. For example, in `frog`, the accuracy of Top1Sim drops to 84% as $\sigma_{cf} = 0.2$, while the accuracy of FedSim remains 91%.

The results of real-world datasets are summarized in Table 1. We also calculate the estimated improvement $\Delta(Top1Sim)$ (denoted as Δ for simplicity) according to our proposed metric. The relationship between $\Delta(Top1Sim)$ and the relative improvement on Top1Sim is presented in Figure 5(ii). Three observations can be made from the results. First, FedSim consistently produces the best performance on all five datasets, while the baselines (especially the separated approaches) only have good performances on specific datasets. For example, Top1Sim has close performance to FedSim on `game`, but fails on `bike`; AvgSim has close performance to FedSim on `bike`, but fails on `game`. Second, $\Delta(Top1Sim)$ is positively correlated with real improvement on Top1Sim, indicating that the metric can be effectively used to estimate the improvement of FedSim without training. This also implies that FedSim can effectively reduce the effect of lost information and redundant information as expected. Third, comparing the performance of removing each component from FedSim, the sort gate makes the most significant contribution to the performance of FedSim by stabilizing the updates of the merge model. The improvement of the weight gate indicates that

adjusting the distribution of similarities can slightly benefit the performance. Besides, CNN merge gate can also slightly improve on MLP merge gate by reducing overfitting.

As elaborated in Section D, FedSim only boosts performance on the datasets satisfying a **widely held** assumption: the similarity between identifiers is related to the similarity between data records. Our experiments in Appendix F.6 indicate that FedSim has close (small K) or lower (large K) performance compared to baselines on a synthetic dataset with independent identifiers.

6.3 Privacy

In this subsection, to study how additional noise on similarities affects the performance of FedSim, we conduct experiments on five real-world datasets (the result of `company` is included in Appendix F.1 due to page limit). Specifically, string or numeric identifiers are converted to Bloom filters according to [29]. The Hamming distances between Bloom filters are used to calculate raw similarities. Then, given an acceptable success rate τ, a noise scale σ is calculated according to Theorem 2. Finally, Gaussian noise with scale σ is added to the raw similarities according to Equation 2. The results are presented in Figure 6. `Exact` is not evaluated since few Bloom filters have exactly the same bits. From Figure 6, we observe that FedSim is robust to the noise on similarities; therefore, the attacking success rate can be reduced to $[10^{-4}, 10^{-3}]$ without evident performance loss. Notably, the performance when $\tau = 1$ is not necessarily the same as the performance in Section 6.2 since similarities are calculated based on different distances.

7 Related Work

Most studies [18, 26, 37, 50, 56, 61] in VFL focus on training and simply assumes record linkage has been done (i.e., the implicit exact linkage on record ID), which is impractical since most real-world federated datasets are unlinked. Some approaches exactly link the identifiers by exact PPRL [7] or private set intersection (PSI) [7, 36, 60]. Nonetheless, these approaches incur performance loss of VFL and are also impractical since the common features of many real-world federated datasets cannot be exactly linked (e.g. GPS location). [22] greedily links the most similar identifiers in PPRL, which negatively impacts performance since some beneficial pairs with relatively low similarity may be neglected. [39, 40] explore the impact of record linkage on the performance of VFL, which is also adopted by [27]. However, all of them focus only on the most similar identifiers and assume there is a one-to-one mapping between the data records of two parties, which is not always true in practice.

Current VFL frameworks support various machine learning models including linear regression [17], logistic regression [25], support vector machine [34], gradient boosting decision trees [7, 60]. FDML [26] supports neural networks but it requires all the parties to hold labels. SplitNN [56] focuses on neural networks and provides a new idea of collaborative learning where the model is split and held by multiple parties. Since we study the scenario where only one party holds the labels and want to support commonly used neural networks, we build FedSim on top of SplitNN.

8 Conclusion

In this paper, we propose FedSim, a novel VFL framework based on similarities to boost the performance of VFL by directly utilizing the similarities calculated in PPRL and skipping the classification process. We also theoretically analyze the additional privacy risk introduced by sharing similarities and provide a bound for the success rate of an intuitive attack. In our experiment, FedSim consistently outperforms all the baselines.

Acknowledgements

This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG2-RP-2020-018). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of National Research Foundation, Singapore. Qinbin is also in part supported by a Google PhD Fellowship.

References

[1] Airbnb. Airbnb prices. `http://insideairbnb.com/get-the-data.html`, 2019.

[2] Manfred Antoni and Arne Bethmann. Pass-befragungsdaten verknüpft mit administrativen daten des iab, 2014.

[3] Manfred Antoni, Ernst Maug, and Stefan Obernberger. Private equity and human capital risk. *Journal of Financial Economics*, 133(3):634–657, 2019.

[4] Manfred Antoni and Rainer Schnell. The past, present and future of the german record linkage center (grlc). *Jahrbücher für Nationalökonomie und Statistik*, 239(2):319–331, 2019.

[5] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403. PMLR, 2018.

[6] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset, 2011.

[7] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, and Qiang Yang. Secureboost: A lossless federated learning framework. *arXiv*, 2019.

[8] CitiBike. Citi bike system data. `https://www.citibikenyc.com/system-data`, 2016.

[9] Michela Coppola, Bettina Lamla, et al. Saving and old age provision in germany (save). design and enhancements. *Schmollers Jahrbuch*, 133(1):109–16, 2013.

[10] David Cournapeau. Sklearn API: make_classification, 2021.

[11] Nik Davis. Steam store games (clean dataset). `https://www.kaggle.com/nikdavis/steam-store-games`, 2019.

[12] Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. *arXiv*, 2016.

[13] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[14] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[15] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[16] Johanna Eberle and Michael Weinhardt. Record linkage of the linked employer-employee survey of the socio-economic panel study (soep-lee) and the establishment history panel (bhp). *German Record Linkage Center Working Paper Series*, 2016.

[17] Siwei Feng and Han Yu. Multi-participant multi-class vertical federated learning. *arXiv*, 2020.

[18] Fangcheng Fu, Yingxia Shao, Lele Yu, Jiawei Jiang, Huanran Xue, Yangyu Tao, and Bin Cui. Vf^{2}boost: Very fast vertical federated gradient boosting for cross-enterprise learning. In *SIGMOD*, 2021.

[19] Daniel Fuß, Jutta von Maurice, and Hans-Günther Roßbach. A unique research data infrastructure for educational research and beyond: The national educational panel study. *Jahrbücher für Nationalökonomie und Statistik*, 236(4):517–528, 2016.

[20] Tobias Gramlich. 'strokes'–record linkage der schlaganfälle in hessen 2007-2010 (strokes-record linkage of stroke cases in hesse 2007-2010). *Available at SSRN 3549212*, 2014.

[21] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[22] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, and et al. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv*, 2017.

[23] Singapore HDB. Resale flat prices in Singapore. `https://data.gov.sg/dataset/resale-flat-prices`, 2018.

[24] Trung Hoang. Video game dataset. `https://www.kaggle.com/jummyegg/rawg-game-dataset`, 2020.

[25] Yaochen Hu, Peng Liu, Linglong Kong, and Di Niu. Learning privately over distributed features: An admm sharing approach. *arXiv*, 2019.

[26] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. Fdml: A collaborative machine learning framework for distributed features. In *KDD*, 2019.

[27] Yan Kang, Yang Liu, and Tianjian Chen. Fedmvt: Semi-supervised vertical federated learning with multiview training. *arXiv*, 2020.

[28] Alexandros Karakasidis, Georgia Koloniari, and Vassilios S Verykios. Scalable blocking for privacy preserving record linkage. In *KDD*, 2015.

[29] Dimitrios Karapiperis, Aris Gkoulalas-Divanis, and Vassilios S Verykios. Federal: A framework for distance-aware privacy-preserving record linkage. *TKDE*, 2017.

[30] Dimitrios Karapiperis and Vassilios S Verykios. An lsh-based blocking approach with a homomorphic matching technique for privacy-preserving record linkage. *TKDE*, 2014.

[31] Martin Kroh, Simon Kühne, Jan Goebel, and Friederike Preu. The 2013 iab-soep migration sample (m1): Sampling design and weighting adjustment. Technical report, SOEP Survey Papers, 2015.

[32] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *TKDE*, 2021.

[33] Yang Liu, Yan Kang, Chaoping Xing, Tianjian Chen, and Qiang Yang. A secure federated transfer learning framework. *IEEE Intelligent Systems*, 2020.

[34] Yang Liu, Yan Kang, Xinwei Zhang, Liping Li, Yong Cheng, Tianjian Chen, Mingyi Hong, and Qiang Yang. A communication efficient collaborative learning framework for distributed features. *arXiv*, 2019.

[35] Yang Liu, Yingting Liu, Zhijie Liu, Yuxuan Liang, Chuishi Meng, Junbo Zhang, and Yu Zheng. Federated forest. *IEEE Transactions on Big Data*, 2020.

[36] Yang Liu, Xiong Zhang, and Libin Wang. Asymmetrically vertical federated learning. *arXiv*, 2020.

[37] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. Feature inference attack on model predictions in vertical federated learning. In *ICDE*, 2021.

[38] Kevin P Murphy. Conjugate bayesian analysis of the gaussian distribution, 2007.

[39] Richard Nock, Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, and et al. The impact of record linkage on learning from feature partitioned data. In *International Conference on Machine Learning*. PMLR, 2021.

[40] Richard Nock, Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Entity resolution and federated learning get a federated resolution. *arXiv*, 2018.

[41] Anja Perry and Beatrice Rammstedt. The research data center piaac at gesis. *Jahrbücher für Nationalökonomie und Statistik*, 236(5):581–593, 2016.

[42] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[43] Zhongang Qi, Saeed Khorram, and Fuxin Li. Visualizing deep networks by optimizing with integrated gradients. In *CVPR Workshops*, volume 2, 2019.

[44] Qichen Qiu. Kaggle dataset: housing price in Beijing. https://www.kaggle.com/ruiqurm/lianjia, 2017.

[45] Salary.sg. Secondary school rankings in Singapore. https://www.salary.sg/2020/secondary-schools-ranking-2020-psle-cut-off/, 2020.

[46] Rory Sayres, Ankur Taly, Ehsan Rahimy, Katy Blumer, David Coz, Naama Hammel, Jonathan Krause, Arunachalam Narayanaswamy, Zahra Rastegar, Derek Wu, et al. Using a deep learning algorithm and integrated gradients explanation to assist grading for diabetic retinopathy. *Ophthalmology*, 2019.

[47] U.S. Small Business Administration (SBA). Loan transactions between companies and banks in US. https://www.kaggle.com/datasets/mirbektoktogaraev/should-this-loan-be-approved-or-denied, 2019.

[48] Christopher-Johannes Schild. Linking'orbis' company data with establishment data from the german federal employment agency. *NO. WP-GRLC-2016-02*, 2016.

[49] Christopher-Johannes Schild and Manfred Antoni. Linking survey data with administrative social security data-the project'interactions between capabilities in work and private life'. *German Record Linkage Center, Working Paper Series, No. WP-GRLC-2014-02*, 2014.

[50] Shreya Sharma, Chaoping Xing, and et al. Secure and efficient federated transfer learning. In *2019 IEEE International Conference on Big Data (Big Data)*, 2019.

[51] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *ICML*, 2017.

[52] New York TLC. TLC trip record data. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, 2016.

[53] Dinusha Vatsalan and Peter Christen. Scalable privacy-preserving record linkage for multiple databases. In *CIKM*, 2014.

[54] Dinusha Vatsalan and Peter Christen. Privacy-preserving matching of similar patients. *Journal of biomedical informatics*, 2016.

[55] Dinusha Vatsalan, Ziad Sehili, Peter Christen, and Erhard Rahm. Privacy-preserving record linkage for big data: Current approaches and research challenges. In *Handbook of Big Data Technologies*. Springer, 2017.

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

[57] Praneeth Vepakomma, Abhishek Singh, Otkrist Gupta, and Ramesh Raskar. Nopeek: Information leakage reduction to share activations in distributed deep learning. *arXiv*, 2020.

[58] Michael Weinhardt, Alexia Meyermann, Stefan Liebig, and Jürgen Schupp. The linked employer–employee study of the socio-economic panel (soep-lee): content, design and research potential. *Jahrbücher für Nationalökonomie und Statistik*, 237(5):457–467, 2017.

[59] Song WenJie and Shen Xuan. Vertical federated learning based on dfp and bfgs. *arXiv*, 2021.

[60] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, and et al. Privacy preserving vertical federated learning for tree-based models. *VLDB*, 2020.

[61] Zhaomin Wu, Qinbin Li, and Bingsheng He. Practical vertical federated learning with unsupervised representation learning. *IEEE Transactions on Big Data*, 2022.

[62] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *TIST*, 2019.

[63] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, and et al. Song. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv*, 2019.

[64] Aakash Kishore Chotrani Zelalem Getahun, Agata. 7+ Million Company Dataset. https://www.kaggle.com/datasets/peopledatalabssf/free-7-million-company-dataset, 2019.

[65] Mingkai Zheng, Shan You, Lang Huang, Fei Wang, Chen Qian, and Chang Xu. Simmatch: Semi-supervised learning with similarity matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14471–14481, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See the last paragraph of Section 1.
   (b) Did you describe the limitations of your work? [Yes] See Appendix G.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix H.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 5.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The codes are included in the supplementary materials. The data are all publicly available; the link or citations of data are presented in Appendix E.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All the details including data splits and hyperparameters are introduced in Section 6.1 and Appendix E.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The variance is reported in all tables and figures.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the "hardware" paragraph of Appendix E.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 6.1 and Appendix E.
   (b) Did you mention the license of the assets? [Yes] See the "license" paragraph of Appendix E.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our codes are included in supplementary materials.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See the "Consent of dataset" paragraph of Appendix H
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See the "Personally identifiable information" of Appendix H.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] Our research is not related to human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] Our research is not related to human subjects.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] Our research is not related to human subjects.