ReMisVFU: Vertical Federated Unlearning via Representation Misdirection for Intermediate Output Feature

Wenhan Wu[1,*] , Zhili He[1,2,*], Huanghuang Liang[1], Yili Gong[1], Jiawei Jiang[1], Chuang Hu[1,3,†], Dazhao Cheng[1,†]

[1]School of Computer Science, Wuhan University, Wuhan, China
[2]Southwest Aluminium (GROUP) Co., Ltd., ChongQing, China
[3]Key Laboratory of Transportation Internet of Things, Wuhan University of Technology, Wuhan, China
{wenhanwu, 2022182110069, hhliang, yiligong, jiawei.jiang, handc, dcheng}@whu.edu.cn

Abstract

Data-protection regulations such as the GDPR grant every participant in a federated system a right to be forgotten. Federated unlearning has therefore emerged as a research frontier, aiming to remove a specific party's contribution from the learned model while preserving the utility of the remaining parties. However, most unlearning techniques focus on Horizontal Federated Learning (HFL), where data are partitioned by samples. In contrast, Vertical Federated Learning (VFL) allows organizations that possess complementary feature spaces to train a joint model without sharing raw data. The resulting feature-partitioned architecture renders HFL-oriented unlearning methods ineffective. In this paper, we propose ReMisVFU, a plug-and-play representation misdirection framework that enables fast, client-level unlearning in splitVFL systems. When a deletion request arrives, the forgetting party collapses its encoder output to a randomly sampled anchor on the unit sphere, severing the statistical link between its features and the global model. To maintain utility for the remaining parties, the server jointly optimizes a retention loss and a forgetting loss, aligning their gradients via orthogonal projection to eliminate destructive interference. Evaluations on public benchmarks show that ReMisVFU suppresses back-door attack success to the natural class-prior level and sacrifices only about 2.5% points of clean accuracy, outperforming state-of-the-art baselines.

Introduction

Federated Learning (FL) (McMahan et al. 2017) allows multiple clients to train a shared model while keeping raw data local, exchanging only encrypted gradients or intermediate representations. Existing mainstream paradigm of FL is horizontal federated learning (HFL) (Yang et al. 2020), which assumes all parties own data drawn from the same feature space, yet different samples. However, this design struggles when collaborating organizations possess heterogeneous feature spaces. To address the limitation, Vertical Federated Learning (VFL) (Cheng et al. 2021; Liu et al. 2024a) was proposed. The VFL parties refer to the same set of users, but each controls a disjoint subset of features. Given a typical cross-industry scenario, an e-commerce platform that holds customers' purchase histories and a bank that records their income and credit scores can train a credit-scoring model without revealing their respective raw data via VFL, thereby reducing privacy-leakage risk.

Recently, with the enforcement of the EU GDPR (Voigt and Von dem Bussche 2017) and the California CCPA (Bonta 2022), the *right to be forgotten*, i.e., the ability to erase specific data from an ML system, has become a key legal requirement. This mandate has spurred research on *Federated Unlearning (FU)* (Liu et al. 2020; Varshney, Vandikas, and Torra 2025; Liu et al. 2024b; Wu et al. 2025c), the federated counterpart of machine unlearning. FU seeks to adjust an already-trained model so that, once a user's erasure request is honored, the resulting model behaves as if that data had never been seen. Although retraining the model from scratch would technically meet the requirement, it is prohibitively time and resource-intensive. Consequently, HFL unlearning (HFU) research has focused on more efficient alternatives, such as reusing stored gradients (Liu et al. 2021, 2022), applying knowledge distillation (Zhu, Li, and Hu 2023; Wu et al. 2025b; Xie et al. 2024) to remove or preserve information selectively, and performing gradient-ascent updates that directly counteract the departing client's contribution (Pan et al. 2025; Hua, Xia, and Xu 2024; Li et al. 2023). However, nearly all existing techniques targeting HFL cannot be transferred to VFL without major redesign because the two paradigms differ fundamentally in both training and inference. State-of-the-art VFL systems, such as splitVFL (Gupta and Raskar 2018), partition a model into multiple local (bottom) models hosted by passive parties (feature holders) and a global (top) model hosted by the active party (label holder). During training, each bottom model forwards intermediate output features to the top model and then completes forward and back-propagation; inference likewise requires all parties to collaborate. Therefore, we identify two unique challenges in VFL unlearning:

- **C1. Data is vertically partitioned**: Unlike HFU, which directly removes entire labeled samples, VFU targets only the feature slice contributed by the departing party,

[*] These authors are co-first authors.

[†] These authors are co-corresponding authors.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

whose influence on prediction is more concealed within the model and more difficult to eliminate completely.

- **C2. Model partitioning and collaborative inference**: A departing party's features affect both its local bottom model and the active party's top model, so unlearning must revise both components and reorganize the inference workflow to fit the updated VFL architecture.

To tackle these challenges, we propose a novel VFU framework named REMISVFU (**Re**presentation **Mis**direction for **V**ertical **F**ederated **U**nlearning). Instead of retraining the model from scratch, REMISVFU performs representation-level misdirection: the forgetting party replaces its intermediate features with anchor vectors sampled from a uniform sphere, which retains the information flow from the departing party while preserving the original VFL pipeline. To ensure utility preservation, we jointly optimize forgetting and retention objectives using a gradient projection strategy that resolves conflicts between their respective gradients. Our key contributions are summarized as follows:

- We propose a representation misdirection mechanism that effectively erases the statistical contribution of the forgetting party by collapsing its output features.
- We design a coordination strategy that ensures retention gradients are orthogonal to forgetting gradients, preventing mutual interference and enabling stable optimization.
- We develop a plug-and-play unlearning pipeline fully compatible with existing splitVFL systems, requiring no changes to communication patterns or party workflows.
- Comprehensive evaluations demonstrate that REMISVFU achieves strong performance both on forgetting and retention objectives.

Preliminary and Related Works

Vertical Federated Learning

VFL departs fundamentally from the HFL paradigm. In HFL every client retains a local subset of samples that share an identical feature space. In VFL, by contrast, each party owns a subset of features describing an overlapping subset of the same individuals. We first formalize the VFL setting adopted in the paper. We denote a dataset with N samples:

$$\mathcal{D} = \big\{(x_i, y_i)\big\}_{i=1}^{N}, \qquad x_i \in \mathbb{R}^{1\times d},\ y_i \in \mathcal{Y}, \tag{1}$$

VFL aims to collaboratively train a global model M while preserving each party's data privacy and model security. Formally, we solve the above optimization problem:

$$\min_{\Omega}\ \mathcal{L}(\Omega;\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} f\big((x_i,y_i);\Omega\big) + \lambda\sum_{k=1}^{K}\gamma(\Omega_k), \tag{2}$$

where $\Omega = \{\Omega_1, \dots, \Omega_K\}$ collects all trainable parameters, $f(\cdot)$ is the task loss (e.g. cross-entropy for classification task), $\gamma(\cdot)$ is a regulariser, and hyperparameter $\lambda > 0$ balances the two terms. Each sample's feature vector is columnwise split among K parties $P_1, \dots, P_K$:

$$x_i = \big[\, x_{i,1} \,\|\, x_{i,2} \,\|\, \dots \,\|\, x_{i,K} \big], \tag{3}$$

where $x_{i,k} \in \mathbb{R}^{1\times d_k}$ and $\sum_{k=1}^{K} d_k = d$. d_k is the feature dimension of party $k \in [1, K]$. One designated *active* party—without loss of generality P_K—additionally retains the label, i.e. $y_i = y_{i,K}$. Thus the local dataset for the passive party $P_k, k \in [1, K-1]$ is:

$$\mathcal{D}_k = \{x_{i,k}\}_{i=1}^{N}, \quad k \in [1, K-1], \tag{4}$$

and the local dataset for the active party P_K is:

$$\mathcal{D}_K = \big\{(x_{i,K}, y_{i,K})\big\}_{i=1}^{N}. \tag{5}$$

Then we can decompose the global model into bottom models $\mathrm{ML}_k(\cdot;\omega_k)$ hosted on P_k ($k = 1, \dots, K$) and a top module $\mathrm{MG}_K(\cdot;\varphi_K)$ that resides only on the active party. With $\Omega_k = \omega_k$ for $k < K$ and $\Omega_K = (\omega_K, \varphi_K)$, the per-sample loss can be rewritten as:

$$\begin{aligned} &f\big((x_i,y_i);\Omega\big) \\ =&\ell\Big(\mathrm{MG}_K\big(\mathrm{ML}_1(x_{i,1};\Omega_1), \dots, \mathrm{ML}_K(x_{i,K};\Omega_K);\varphi_K\big),\ y_{i,K}\Big), \end{aligned} \tag{6}$$

where $\ell(\cdot,\cdot)$ denotes the task-specific loss. Eq. (6) assumes that the active party P_K owns a subset of features $x_{i,K}$ in addition to the labels $y_{i,K}$. However, in many real-world deployments, the entity holding the labels is not a data provider but merely an annotator (e.g., a hospital supplying diagnoses, or a third-party rating agency). In this label-only setting, we simply let $d_K = 0$ so that $x_{i,K} = \varnothing$ and the active party contributes no local encoder ML_K. Consequently, $f\big((x_i,y_i);\Omega\big)$ in Eq. (6) reduces to:

$$\ell\Big(\mathrm{ML}_1(x_{i,1};\Omega_1), \dots, \mathrm{ML}_{K-1}(x_{i,K-1};\Omega_{K-1});\varphi_K\big),\ y_{i,K}\Big), \tag{7}$$

where MG_K aggregates only the encrypted representations received from the passive parties. A VFL framework accommodates both feature-rich and label-only active parties. The training round proceeds as follows: First, the parties privately align entity IDs (Zhao et al. 2024b). Each party then performs a local forward pass and obtains its immediate hidden representation $h_{i,k} = \mathrm{ML}_k(x_{i,k};\Omega_k)$. Passive parties P_k encrypt (or otherwise protect) $\{h_{i,k}\}$ and transfer them to the active party P_K. The active party decrypts and aggregates the representations through MG_K, evaluates the loss using Eq. (6), computes gradients, and updates φ_K together with all Ω_k. The refreshed Ω_k ($k < K$) are finally returned to their respective owners. Repeating these VFL rounds until convergence achieves joint training while ensuring that raw features and labels remain local. For the model architecture, When deep models are employed, the global module MG_K usually comprises trainable network layers, in a manner reminiscent of vertical SplitNN (Vepakomma et al. 2018; Ceballos et al. 2020), where the entire model is partitioned layer-wise across different parties. This configuration is commonly named splitVFL (Gupta and Raskar 2018), one of the most popular VFL architectures. In this paper, we focus on splitVFL as the primary object of study.

Vertical Federated Unlearning

For the vertical federated unlearning (VFU) problem, the goal is to remove the influence of designated feature owners from an already trained model while preserving the utility for all remaining parties. (Deng et al. 2023) were the first to articulate the concept, though their method is restricted to logistic-regression models. Similarly, (Li et al. 2024a) designed a VFU framework tailored to federated gradient-boosted decision trees. For the deep learning model, (Wang et al. 2024) enables rapid retraining in VFL by retaining checkpoints of lower-layer models, but at the cost of significant storage overhead. (Varshney, Vandikas, and Torra 2025) adopts a knowledge-distillation approach that focuses on changing the model architecture rather than truly removing the memorized information. Building on these studies, we formulate the client-level VFU process as follows:

Let P_f be the party that requests its contributions to be forgotten. Its local data are denoted by $\mathcal{D}_f = \{x_{i,f}\}_{i=1}^{N}$, where $x_{i,f} \in \mathbb{R}^{1\times d_f}$ and $\sum_{k\neq f} d_k + d_f = d$. The remaining parties $\mathcal{P}_r = \{P_k\}_{k\neq f}$ with data $\mathcal{D}_r = \{x_{i,k}\}_{k\neq f, i=1}^{N}$ constitute the retained side. It is worth emphasizing that, irrespective of whether the forgetting party P_f acts as the active party, our study deliberately excludes the erasure of label data, because removing the labels would strip the model of its supervisory signal and thus preclude further execution of the classification task.

Baseline (retrained) model. Ideally, the system would discard all parameters related to $\mathcal{D}_f$ and retrain splitVFL from scratch using only the retained parties:

$$\Omega^{\star} \;=\; \mathcal{VFL}\big(\mathcal{D} \setminus \mathcal{D}_f\big), \tag{8}$$

where $\mathcal{VFL}$ denotes the standard training pipeline described in Figure **??**. Model $\Omega^{\star}$ serves as the *gold standard* for evaluating unlearning quality (Wang et al. 2022; Meerza, Sadovnik, and Liu 2024).

Unlearning process. In practice, it is often impractical to perform full retraining because of the high resource cost. We therefore introduce a VFU operator as follows:

$$\mathcal{VU}:\; \mathcal{VFL}(\mathcal{D}) \;\otimes\; \mathcal{D}_r \;\otimes\; \mathcal{D}_f \;\longrightarrow\; \widetilde{\Omega}, \tag{9}$$

which takes the original splitVFL model $\mathcal{VFL}(\mathcal{D})$, the complete retained data $\mathcal{D}_r$ (only locally available at each party), and the forgetting partition $\mathcal{D}_f$ as input, and outputs an unlearned parameter set $\widetilde{\Omega}$.

Unlearning objective. VFU is successful when the distribution of predictions made by $\widetilde{\Omega}$ is approximately indistinguishable from that of the baseline model in Eq. (8):

$$\Phi\big[\mathcal{VFL}(\mathcal{D} \setminus \mathcal{D}_f); \Omega^{*}\big] \;\approx\; \Phi\big[\mathcal{VU}\big(\mathcal{VFL}(\mathcal{D}), \mathcal{D}, \mathcal{D}_f\big); \widetilde{\Omega}\big], \tag{10}$$

where $\Phi[\cdot]$ denotes the predictive distribution of a model. Kullback–Leibler divergence can quantify the discrepancy between the two distributions (Kullback and Leibler 1951):

$$\mathrm{KL}\big(\Phi^{\star} \parallel \widetilde{\Phi}\big) \;=\; \mathbb{E}_{x\sim\Phi^{\star}}\Big[\log \Phi^{\star}(x)/\widetilde{\Phi}(x)\Big]\,, \tag{11}$$

where $\Phi^{\star} = \Phi\big[\mathcal{VFL}(\mathcal{D} \setminus \mathcal{D}_f)\big]$ and $\widetilde{\Phi} = \Phi\big[\widetilde{\Omega}\big]$. An effective unlearning algorithm seeks to minimize Eq.(11) with minimal resource and time overhead.

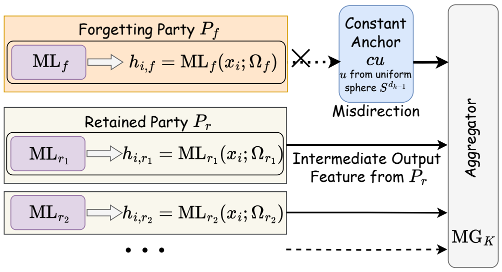

Forgetting Party P_f
ML_f
$h_{i,f} = \mathrm{ML}_f(x_i; \Omega_f)$
Constant Anchor
cu
u from uniform sphere $S^{d_{h-1}}$
Misdirection
Retained Party P_r
ML_{r_1}
$h_{i,r_1} = \mathrm{ML}_{r_1}(x_i; \Omega_{r_1})$
Intermediate Output Feature from P_r
ML_{r_2}
$h_{i,r_2} = \mathrm{ML}_{r_2}(x_i; \Omega_{r_2})$
Aggregator
MG_K

Figure 1: Representation Misdirection Unlearning in VFL.

Methodology

System Overview

REMISVFU is a plug-and-play unlearning module for splitVFL. When the forgetting party P_f submits an erasure request, we apply representation misdirection to the intermediate output features produced by its bottom model/encoder ML_f, which means P_f substitutes its informative representation $h_{i,f}$ with a target anchor derived from a random vector. This procedure removes the influence of P_f's representations on the global model while requiring only a few additional training epochs. The active party P_K harmonizes the resulting gradients between the retained gradients and forgotten gradients, and all parameters Ω are updated collaboratively, achieving fast erasure with virtually no significant drop in overall accuracy.

We study a classification-oriented deep VFL architecture that concatenates the intermediate feature representations produced by multiple parties and feeds them into a top aggregator MG_k to generate the final prediction. Our objective is twofold: (i) suppress the model's capacity to extract information originating from the forgotten party P_f, and (ii) simultaneously preserve its ability to exploit the information contributed by the retained parties P_r $(r \neq k)$.

For the unlearning procedure, we depart from conventional class-level unlearning that targets accuracy metrics directly (Chundawat et al. 2023; Chen et al. 2023). Instead, we pursue a more generalizable approach to erase the entire knowledge distribution contributed by P_f. Each VFU round reuses the exact communication pattern of standard VFL, leaving the workflow of passive parties $P_{k\neq f}$ completely unaffected; hence, the method can be integrated into existing VFL frameworks without any protocol modification.

Representation Misdirection (Forgetting) Loss $\mathcal{L}_f$

Traditionally, deep neural networks are trained by minimizing a loss that depends solely on their final outputs. Mechanistic interpretability proposes editing a model by intervening on individual neurons (Mazzia et al. 2024; Mitchell et al. 2021). Departing from both perspectives, we adopt the view that a model's internal representations already encode structured world knowledge and can therefore be deliberately manipulated to steer the model's behavior (Wang et al. 2023; Li et al. 2024b; Wu et al. 2025a). In VFL, the intermediate features emitted by each party naturally serve as such representations and can be harnessed for unlearning. Building on this insight, we devise a representation-misdirection forgetting loss that injects controlled perturbations into the features contributed by the forget party P_f based on random alignment. Intuitively, this loss scrambles the intermediate outputs of P_f so thoroughly that the top aggregator MG_K can no longer recover any useful information from that party, which achieves the unlearning objectives.

More specifically, as shown in Figure 1, to compute the forgetting loss, we have access to $h_{i,f}$, i.e., the latent feature representations $\mathrm{ML}_f(x_{i,f};\Omega_f)$ produced by the forget party P_f's bottom model for an input sample $x_{i,f}$. To obliterate the statistical footprint of $\mathcal{D}_f$, we draw once a unit vector $\mathbf{u} \sim \mathcal{U}(\mathbb{S}^{d_h-1})$ and force every latent vector $h_{i,f}$ towards the constant anchor $c\mathbf{u}$. c is a hyperparameter that controls the scaling of the intermediate output features. d_h denotes the dimensionality of the intermediate representation, i.e., $d_h = \dim\,(h_{i,f})$. We define:

$$\mathbb{S}^{d_h-1} = \left\{\mathbf{v} \in \mathbb{R}^{d_h} \mid \|\mathbf{v}\|_2 = 1\right\}, \tag{12}$$

the unit $(d_h - 1)$-sphere in $\mathbb{R}^{d_h}$, comprising all vectors with Euclidean norm 1. Let $\mathcal{U}(\mathbb{S}^{d_h-1})$ be the uniform distribution over this sphere. Thus, sampling $\mathbf{u} \sim \mathcal{U}(\mathbb{S}^{d_h-1})$ means drawing a unit vector whose direction is uniformly distributed across all possible orientations while its length remains exactly 1. Once drawn, $\mathbf{u}$ is kept fixed for the entire training process of the designated forget party P_f. Given the forget dataset $\mathcal{D}_f$, the squared-ℓ_2 loss is as below:

$$\mathcal{L}_f = \mathbb{E}_{x_{i,f}\sim\mathcal{D}_f}\left\|h_{i,f} \;-\; c\mathbf{u}\right\|_2^2, \tag{13}$$

$\mathcal{L}_f$ quickly collapses the encoder's output manifold, guaranteeing that no membership or attribute information about P_f's data can be recovered from the new representation.

Utility Preservation (Retention) Loss $\mathcal{L}_r$

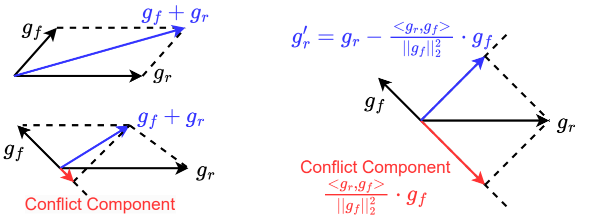
The retention aim is to curb the general capability loss incurred by unlearning. While $\mathcal{L}_f$ enforces forgetting, a concurrent utility loss $\mathcal{L}_r$ safeguards the service quality for the remaining parties. Rather than minimizing surrogate objectives such as KL divergence, we directly optimize the task-native loss. We reuse the original splitVFL forward path and compute the retention loss as:

$$\mathcal{L}_r = \mathbb{E}_{(x_{i,1},\ldots,x_{i,K},\,y_{i,K})\sim\mathcal{D}}\;\ell\Big(\mathrm{MG}_K\big(\{h_{i,k}\}_{k=1}^{K}\big),\;y_{i,K}\Big), \tag{14}$$

where $\ell(\cdot)$ is either cross-entropy loss for the classification tasks or MSE loss for the regression tasks.

Coordinating Gradients for Joint Optimization

The preceding sections introduced the forgetting loss $\mathcal{L}_f$ and the retention loss $\mathcal{L}_r$. Jointly optimizing these objectives is challenging because their gradient directions frequently conflict. Strengthening the knowledge of the retained parties can inadvertently encourage the model to preserve information from the forget set $\mathcal{D}_f$ as shown in Figure 2a, thereby undermining erasure. Conversely, aggressive erasure may trigger catastrophic forgetting of transferable knowledge, ultimately diminishing downstream performance (Zhao et al. 2024a; Choi et al. 2024).


$g_f + g_r$
g_f
g_r
$g_f + g_r$
g_f
g_r
Conflict Component
$g'_r = g_r - \frac{<g_r,g_f>}{||g_f||_2^2} \cdot g_f$
g_f
g_r
Conflict Component
$\frac{<g_r,g_f>}{||g_f||_2^2} \cdot g_f$

(a) Update with Sum Gradients: $g_f + g_r$

(b) Update with Coordinating Gradients: $g_f + g'_r$

Figure 2: Coordinating gradients for joint optimization. (a) Directly summing g_r and g_f may cause interference due to conflicting directions. (b) Projecting out the conflicting component yields g'_r, which is orthogonal to g_f, thus avoiding interference and preserving forgetting effectiveness.

Motivated by insights from multi-task learning (Yu et al. 2020; Chai et al. 2022; Huang, Foo, and Liu 2024), we introduce a gradient–projection coordination strategy that aligns the retention and forgetting objectives in parameter space to mitigate gradient conflicts. By orthogonalizing their gradients, the optimizer follows a coherent trajectory that advances both goals efficiently. The detail is given as shown in Figure 2b. First, we denote the gradients of Eqs. (13) and (14) for forgetting and retention, respectively:

$$g_r \;=\; \nabla_\Omega\,\mathcal{L}_r, \qquad g_f \;=\; \nabla_\Omega\,\mathcal{L}_f, \tag{15}$$

A straightforward approach would update the model with the weighted sum as follows:

$$\nabla_\Omega\mathcal{L}_{all} = \nabla_\Omega\big(\mathcal{L}_f + \alpha\mathcal{L}_r\big) = g_f + \alpha g_r, \tag{16}$$

where $\alpha > 0$ is a trade-off coefficient that scales the retention gradient, balancing preservation of utility against the strength of the forgetting signal. However, when g_r and g_f point in opposing directions, their components may cancel out, making forgetting sluggish and convergence unstable. To overcome the sub-optimality of naive joint optimization, we adopt a gradient-projection strategy. We measure the cosine similarity between g_r and g_f as below:

$$\cos(g_r, g_f) = \frac{\langle g_r, g_f\rangle}{\|g_r\|_2\|g_f\|_2}. \tag{17}$$

If the similarity is negative, the two objectives are locally conflicting. To ensure complete erasure and preclude interference from the retention objective with the forgetting objective, we then project g_r onto the subspace orthogonal to g_f. By stripping g_r of its conflicting component $\frac{\langle g_r, g_f\rangle}{\|g_f\|_2^2}\,g_f$, we obtain the adjusted gradient $\widetilde{g}_r$ as follows:

$$\widetilde{g}_r = \begin{cases} g_r - \dfrac{\langle g_f, g_r\rangle}{\|g_f\|_2^2}\,g_f, & \langle g_f, g_r\rangle < 0, \\ g_r, & \text{otherwise}, \end{cases} \tag{18}$$

This orthogonality removes gradient conflicts, so updating with the projected gradient $\widetilde{g}_r$ incurs far less adverse impact on the forgetting objective. Conversely, if the initial cosine

Algorithm 1: ReMisVFU Process

```
Input: Pre-trained VFL model $\Omega = \{\Omega_1, \ldots, \Omega_K, \varphi_K\}$;
       Forgetting party index $f$ $(1 \leq f < K)$ with $\mathcal{D}_f$;
       Retained parties $\mathcal{P}_r = \{1, \ldots, K\} \setminus \{f\}$ with $\mathcal{D}_r$;
       Hyper-parameters $c, \alpha$, learning rate $\eta$, epochs $T$.
Output: Unlearned model $\widetilde{\Omega}$.
Initialize each party with the pretrained parameters $\Omega$ .
Draw once $\mathbf{u} \sim \mathcal{U}(\mathbb{S}^{d_h - 1})$ .                ▷ kept fixed
for $t \leftarrow 1$ to $T$ do
    foreach party $P_k \in \{\mathcal{P}_r, \mathcal{P}_f\}$ in parallel do
        Sample mini-batch $\{x_{i,k}\}_{i=1}^{B}$;
        $h_{i,k} \leftarrow \mathrm{ML}_k(x_{i,k}; \Omega_k)$ .                ▷ local forward
    end
    $P_K$ receives encrypted $\{h_{i,k}\}_{k=1}^{K-1}$ or $\{h_{i,k}\}_{k=1}^{K}$,
      decrypt, concatenate;                ▷ active part aggregation
    $\hat{y}_i \leftarrow \mathrm{MG}_K(\{h_{i,k}\}_{k=1}^{K}; \varphi_K)$ .                ▷ active part forward
    Computing forgetting loss and gradients using Eq.(13):
      $\mathcal{L}_f \leftarrow \frac{1}{B}\sum_{i=1}^{B}\left\|h_{i,f} - c\mathbf{u}\right\|_2^2,\ g_f \leftarrow \nabla_\Omega \mathcal{L}_f$;
    Computing retention loss and gradients using Eq. (14):
      $\mathcal{L}_r \leftarrow \frac{1}{B}\sum_{i=1}^{B}\ell(\hat{y}_i, y_{i,K}),\ g_r \leftarrow \nabla_\Omega \mathcal{L}_r$;
    Coordinating gradients for $g_f$ and $g_r$ using Eq. (18):
      $\widetilde{g}_r \leftarrow g_r - \frac{\langle g_f, g_r \rangle}{\|g_f\|_2^2} g_f$ if $\langle g_f, g_r \rangle < 0$ else $g_r$;
    Updating global model $\Omega$ and broadcast to all the
      parties $\{\mathcal{P}_r, \mathcal{P}_f\}$: $\Omega \leftarrow \Omega - \eta\,(g_f + \alpha\,\widetilde{g}_r)$ .
end
return $\widetilde{\Omega} \leftarrow \Omega$
```

similarity is non-negative, the two gradient vectors are already compatible and no projection is required. The final coordinated gradient update is:

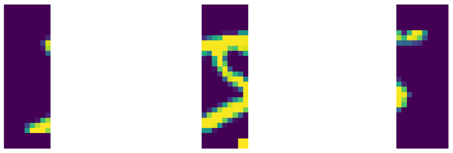
$$\nabla_\Omega \mathcal{L}'_{all} = g_f + \alpha\,\widetilde{g}_r, \tag{19}$$

which retains the full forgetting signal while preserving only the non-interfering component of the retention gradient. Updates based on Eq. (19) continue to drive thorough erasure of $\mathcal{D}_f$ without undermining the model's retention knowledge. Empirically, the projection yields faster convergence and cleaner attainment of the target distribution. The entire process is outlined in Algorithm 1.

Evaluations

Evaluation Setup

Experimental Setup. Our evaluation is conducted on five popular image benchmarks: MNIST (LeCun et al. 1998), Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), SVHN (Netzer et al. 2011), CIFAR-10 and CIFAR-100 (Krizhevsky and Hinton 2009). To emulate a VFL scenario, each image is vertically partitioned into three equal slices (left, centre, and right) as set in papers (Han et al. 2025; Wang et al. 2024). The centre slice is assigned to the target party to be forgotten. Following the SplitNN paradigm, each client P_k employs a local encoder ML_k consisting of two 3×3 convolutional layers (32 and 64 channels, respectively), each followed by a 2×2 max-pooling layer. MG first concatenates all client features, then applies two fully connected layers (ReLU hidden size 128) to generate multi-class prediction logits. Unless otherwise stated, the hyperparameters are fixed to $c = 1.0$ and $\alpha = 10^{-3}$. The experiments are conducted on a machine equipped with an Intel(R) Xeon(R) Gold 6240C CPU, 32GB RAM, and an NVIDIA A100-40GB GPUs.



(a) Left part. (b) Centre Part. (c) Right Part.

Figure 3: An example of partitioning a MNIST image and implanting a backdoor trigger.

Evaluation Metrics. We use 4 evaluation metrics widely used in unlearning: **1) Clean Accuracy**. The standard classification accuracy obtained on a clean test set (free of injected backdoors) measures model utility. **2) Backdoor Accuracy**. To quantify the residual influence of the backdoor and, therefore, the effectiveness of unlearning, we report the proportion of poisoned samples that are classified as the attacker's target label (Han et al. 2025; Nguyen et al. 2024). A lower value, which is ideally close to the natural class prior, indicates more successful unlearning. We adopt a 2×2 white trigger stamped in the lower-right corner and set the target label to "0" as shown in Figure 3. About 10% of the target party's images are poisoned. **3) Membership Inference Attack (MIA).** We further validate unlearning by measuring the success rate of membership inference attacks (MIA) (Li et al. 2025; Kong, Roy Chowdhury, and Chaudhuri 2022). Given the model's output logits with and without the target party, an attacker trains a binary XGBoost classifier (Chen and Guestrin 2016) to decide whether a specific sample participated in VFU. A significant drop in MIA success approaching random guessing after unlearning means effective removal of sensitive information. **4) Runtime Overhead.** This metric captures the runtime cost introduced by the unlearning procedure itself.

Baselines. We compare ReMisVFU with 4 representative baselines: **1) FedRetrain**: fully retrain the VFL model from scratch after removing the target party, which is the "gold standard". **2) FedR2S (Wang et al. 2024)**: a rapid re-training scheme that switches between RAdam and momentum SGD optimisers based on a pre-defined loss threshold. **3) FedGA (Han et al. 2025)**: gradient-ascent based unlearning that cancels the target party's contribution via adversarial gradient updates. **4) FedKD (Varshney, Vandikas, and Torra 2025)**: knowledge-distillation approach that transfers retained knowledge from the old model to a new one while omitting the target data.

Method	MNIST		Fashion-MNIST		SVHN		CIFAR-10		CIFAR-100	
	Clean↑	Bkd.↓	Clean↑	Bkd.↓	Clean↑	Bkd.↓	Clean↑	Bkd.↓	Clean↑	Bkd.↓
Original (Bkd.)	98.69	99.99	91.14	99.71	85.56	98.08	68.12	93.26	36.47	94.35
Original (no Bkd.)	98.89	9.06	91.02	9.91	86.21	9.86	68.52	9.63	36.56	1.91
FedRetrain (Gold)	91.51	9.81	88.30	10.37	70.85	9.64	60.34	10.76	30.39	1.66
FedR2S	89.63	18.47	86.70	18.40	64.89	16.64	54.93	18.07	19.76	27.49
FedGA	88.42	11.09	87.02	11.10	60.74	7.87	59.15	14.01	26.93	22.12
FedKD	87.67	14.35	86.19	12.11	61.15	8.69	59.61	15.17	25.68	26.79
ReMisVFU (Ours)	90.22	9.96	87.08	10.39	67.56	10.22	58.45	10.55	25.72	18.33

Table 1: Clean accuracy (%) and back-door attack success rate (%) on five image benchmarks (closer to FedRetrain is better).

Method	MNIST		Fashion-MNIST		SVHN		CIFAR-10		CIFAR-100	
	AUC.	ACC.	AUC.	ACC.	AUC.	ACC.	AUC.	ACC.	AUC.	ACC.
FedRetrain (Gold)	0.769	0.694	0.616	0.582	0.655	0.620	0.583	0.553	0.618	0.592
FedR2S	0.886	0.857	0.719	0.668	0.691	0.606	0.648	0.579	0.622	0.600
FedGA	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
FedKD	0.887	0.849	0.742	0.658	0.755	0.655	0.750	0.666	0.721	0.657
ReMisVFU (Ours)	0.854	0.832	0.697	0.647	0.684	0.599	0.576	0.525	0.609	0.553

Table 2: Membership inference attack success rate on five image benchmarks (closer to FedRetrain is better).

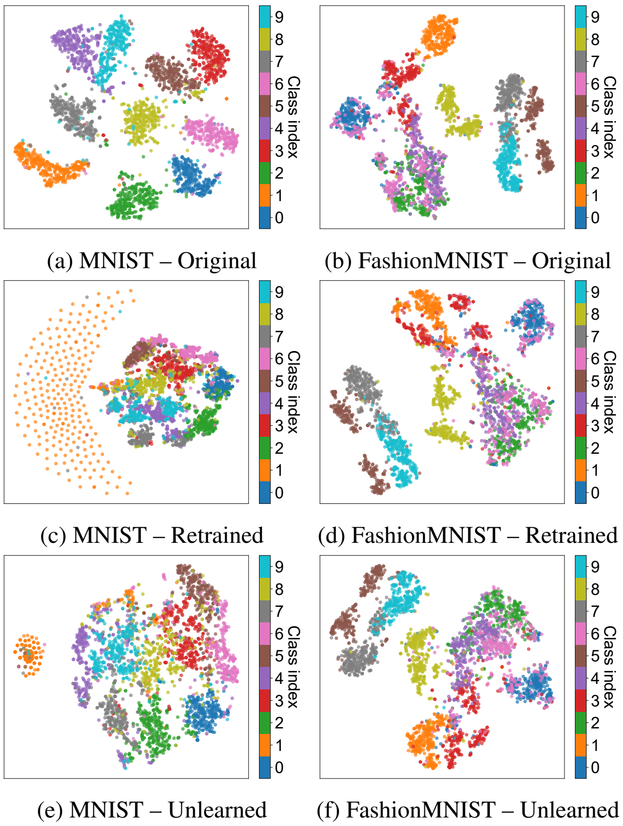
Evaluation Results

Utility and Forgetting Performance. Table 1 compares ReMisVFU with baselines on the joint goals of utility (clean accuracy) and forgetting effectiveness (back-door success rate). Across the 5 datasets, the clean-accuracy of ReMisVFU is only 2.45% lower than the fully retrained "gold" model, while FedR2S, FedGA, and FedKD are on average 5.10%, 3.83%, and 4.22% lower than the retrained model, respectively. In some scenarios, although ReMisVFU is slightly weaker, its privacy performance is superior. Its attack-success rates are closest to the natural class priors in "Original (no Bkd.)" rows and are either the best on every dataset. By contrast, FedR2S leaves substantial back-door residue, increasing it by an average of 7.84%, showing that switching optimizers after loading a pretrained model is insufficient for thorough forgetting. FedGA suppresses triggers to roughly 9–14% on easier datasets but deteriorates to 22.1% on CIFAR-100, while FedKD's performance falls between the FedR2S and FedGA. To sum up, ReMisVFU achieves a more complete erasure of the forgotten features and, combined with its competitive utility, demonstrates the most robust overall performance among all baselines.

Privacy (Membership Inference). We measure post-unlearning privacy leakage with the MIA success rate. The larger the value, the easier it is for an adversary to decide whether a sample participated in training. As is customary, we report AUC (area under the ROC curve) and ACC (attack accuracy); the closer these scores are to FedRetrain, the better the unlearning quality. Table 2 shows that ReMisVFU attains MIA metrics closest to FedRetrain on every dataset: on average, the AUC gap between the two methods is only 4.22% (and 5.82% for ACC). In comparison, FedKD and FedR2S exhibit average AUC gaps of 12.28% and 6.50% (ACC gaps of 8.89% and 5.94%, respectively). FedGA, however, performs catastrophically on all MIA privacy indicators because its gradient-ascent "forgetting" step drives the model into a highly over-fitted, sharp region of the loss landscape.

Visualization. To illustrate the effect of ReMisVFU from a representational viewpoint, we ran t-SNE visualization (Maaten and Hinton 2008) on the intermediate VFL features $\{h_{i,k}\}_{k=1}^{K-1}$ for MNIST and Fashion-MNIST. As shown in Figure 4, after only a few iterations, the resulting feature distribution closely matches that of the fully retrained Gold Baseline, but with a more uniform density. This shows that our representation-misdirection strategy can selectively erase the semantic information contributed by the forgotten party while preserving the discriminative power of the remaining parties, which removes the forgotten party's influence from deeper layers of the model.

Efficiency (Running Overhead). To highlight the runtime efficiency, Figure 5 presents end-to-end unlearning times on each benchmark. ReMisVFU completes forgetting using only 10.63%–21.49% of FedRetrain's time, representing a substantial speed-up. Compared with FedR2S, despite both methods avoiding full retraining, ReMisVFU still cuts wall-clock time by 24.33%, chiefly because it removes the costly optimizer-switching heuristic and attains faster convergence through multi-round feature masking. FedGA also converges more slowly, requiring roughly 3.89× our runtime, as its adversarial ascent steps frequently push the model into high-loss regions, which must later be corrected with additional fine-tuning rounds. FedKD employs a two-stage teacher–student paradigm; the mandatory forward pass of the teacher network in every round incurs extra overhead, resulting a runtime about 1.77× that of ReMisVFU. In summary, by editing internal representations instead of retraining entire networks, ReMisVFU achieves markedly lower time cost.



(a) MNIST – Original (b) FashionMNIST – Original

(c) MNIST – Retrained (d) FashionMNIST – Retrained

(e) MNIST – Unlearned (f) FashionMNIST – Unlearned

Figure 4: t-SNE visualization of the intermediate features $\{h_{i,k}\}_{k=1}^{K-1}$, with different colors denoting different classes.

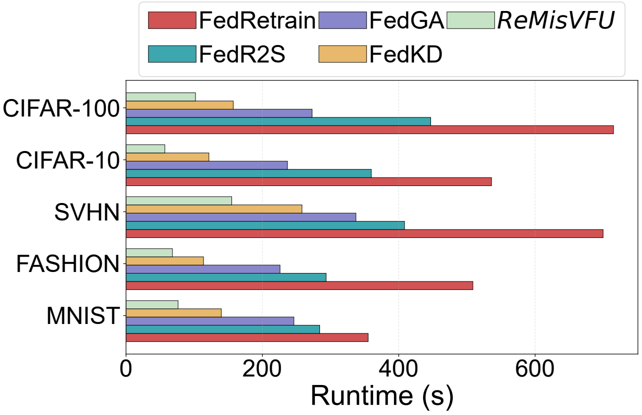
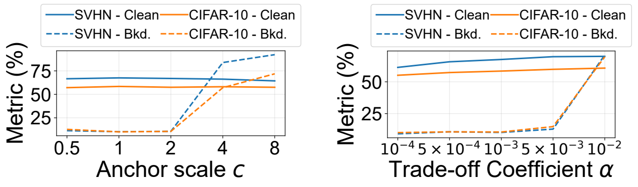


Figure 5: Comparison of the VFU training time.

Method	SVHN		CIFAR-10	
	Clean Acc.↑	Back-door↓	Clean Acc.↑	Back-door↓
No-GCM	65.04±0.85	11.65±1.45	57.54 ±1.24	14.07 ±0.41
Rand-Proj	35.62 ±2.86	82.63 ±3.45	32.07 ±0.19	87.21 ±4.38
ReMisVFU	**67.56** ±1.02	**10.22** ±0.56	**58.45** ±1.17	**10.55** ±0.23

Table 3: Effect of gradient conflict mitigation.



(a) Varying c (b) Varying α

Figure 6: Parameter sensitivity on SVHN and CIFAR-10.

Ablation Study. To quantify the contribution of our gradient–projection strategy in Eq. (18)–(19), we compare ReMisVFU with two degraded variants: i) *No-GCM*, which jointly optimises $\mathcal{L}_f + \alpha\mathcal{L}_r$ using vanilla gradient summation in Eq. (16); ii) *Rand-Proj*, which replaces the orthogonal projection with a random unit vector of the same norm, serving as a sanity check that the benefit indeed stems from conflict resolution rather than mere gradient scaling. Table 3 reports that *No-GCM* raises the back-door attack success rate by 2.47% and lowers the clean accuracy by 1.71% on average, demonstrating that naively summing gradients introduces destructive interference. Random projection further causes a pronounced degradation in both forgotten effectiveness and retained utility. With GCM enabled, our method consistently attains better utility–privacy trade-off, confirming that explicit gradient alignment is essential for VFU.

Parameter Sensitivity Analysis. We investigate the influence of anchor scaling factor c in Eq. (13) and the trade-off coefficient α in Eq. (19), respectively. Figure 6a sweeps $c \in \{0.5, 1, 2, 4, 8\}$ while fixing $\alpha = 10^{-3}$; Figure 6b varies $\alpha \in \{10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}\}$ with $c = 1$. For the anchor scale c, when $c \leq 2$, both clean accuracy and back-door success remain essentially flat, indicating that moderate feature compression does not hinder model utility. Once $c \geq 4$, the degree of forgetting drops sharply, because an excessively large anchor pushes intermediate features into the nonlinear saturation regime of the activation functions, diluting the forgetting signal and allowing the trigger to re-emerge. For the trade-off coefficient α, when α lies in the range $1 \times 10^{-4} \leq \alpha \leq 1 \times 10^{-3}$, clean accuracy rises slightly with increasing α, while the back-door success rate remains near the random-guessing baseline. Once α surpasses 5×10^{-3}, the back-door success on both datasets quickly exceeds 50%, indicating that the retention term overwhelms the forgetting objective. Balancing utility and privacy, we therefore adopt $\alpha = 10^{-3}$ as the default setting.

Conclusion

In this paper, we present ReMisVFU, a plug-and-play vertical federated unlearning framework that cuts off a forgotten party's influence by misdirecting intermediate representations and reconciles the trade-off between forgetting and retention via orthogonal gradient projection. Extensive evaluations show that through this combination, ReMisVFU achieves an efficient balance between privacy-preserving unlearning and strong model utility.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62341410, 62302348), National Key Research and Development Program of China (2023YFE0205700) and the State Key Laboratory of Internet of Things for Smart City (University of Macau) Open Research Project (SKL-IoTSC(UM)/ORP05/2025).

References

Bonta, R. 2022. California consumer privacy act (CCPA). *Retrieved from State of California Department of Justice: https://oag. ca. gov/privacy/ccpa*.

Ceballos, I.; Sharma, V.; Mugica, E.; Singh, A.; Roman, A.; Vepakomma, P.; and Raskar, R. 2020. Splitnn-driven vertical partitioning. *arXiv preprint arXiv:2008.04137*.

Chai, H.; Yin, Z.; Ding, Y.; Liu, L.; Fang, B.; and Liao, Q. 2022. A model-agnostic approach to mitigate gradient interference for multi-task learning. *IEEE Transactions on Cybernetics*, 53(12): 7810–7823.

Chen, M.; Gao, W.; Liu, G.; Peng, K.; and Wang, C. 2023. Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7766–7775.

Chen, T.; and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Cheng, K.; Fan, T.; Jin, Y.; Liu, Y.; Chen, T.; Papadopoulos, D.; and Yang, Q. 2021. Secureboost: A lossless federated learning framework. *IEEE intelligent systems*, 36(6): 87–98.

Choi, D.; Choi, S.; Lee, E.; Seo, J.; and Na, D. 2024. Towards efficient machine unlearning with data augmentation: Guided loss-increasing (gli) to prevent the catastrophic model utility drop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 93–102.

Chundawat, V. S.; Tarun, A. K.; Mandal, M.; and Kankanhalli, M. 2023. Zero-shot machine unlearning. *IEEE Transactions on Information Forensics and Security*, 18: 2345–2354.

Deng, Z.; Han, Z.; Ma, C.; Ding, M.; Yuan, L.; Ge, C.; and Liu, Z. 2023. Vertical federated unlearning on the logistic regression model. *Electronics*, 12(14): 3182.

Gupta, O.; and Raskar, R. 2018. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications*, 116: 1–8.

Han, M.; Zhu, T.; Zhang, L.; Huo, H.; and Zhou, W. 2025. Vertical Federated Unlearning via Backdoor Certification. *IEEE Transactions on Services Computing*.

Hua, Y.; Xia, H.; and Xu, S. 2024. Federated Unlearning for Samples Based on Adaptive Gradient Ascent of Angles. In *2024 IEEE 23rd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 834–841. IEEE.

Huang, M. H.; Foo, L. G.; and Liu, J. 2024. Learning to unlearn for robust machine unlearning. In *European Conference on Computer Vision*, 202–219. Springer.

Kong, Z.; Roy Chowdhury, A.; and Chaudhuri, K. 2022. Forgeability and membership inference attacks. In *Proceedings of the 15th ACM workshop on artificial intelligence and security*, 25–31.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 1(4).

Kullback, S.; and Leibler, R. A. 1951. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1): 79–86.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, B.; Zhang, J.; Li, J.; and Wu, C. 2024a. Securecut: Federated gradient boosting decision trees with efficient machine unlearning. In *International Conference on Pattern Recognition*, 350–365. Springer.

Li, G.; Shen, L.; Sun, Y.; Hu, Y.; Hu, H.; and Tao, D. 2023. Subspace based federated unlearning. *arXiv preprint arXiv:2302.12448*.

Li, N.; Pan, A.; Gopal, A.; Yue, S.; Berrios, D.; Gatti, A.; Li, J. D.; Dombrowski, A.-K.; Goel, S.; Phan, L.; et al. 2024b. The wmdp benchmark: Measuring and reducing malicious use with unlearning. *arXiv preprint arXiv:2403.03218*.

Li, N.; Zhou, C.; Gao, Y.; Chen, H.; Zhang, Z.; Kuang, B.; and Fu, A. 2025. Machine unlearning: Taxonomy, metrics, applications, challenges, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*.

Liu, G.; Ma, X.; Yang, Y.; Wang, C.; and Liu, J. 2020. Federated unlearning. *arXiv preprint arXiv:2012.13891*.

Liu, G.; Ma, X.; Yang, Y.; Wang, C.; and Liu, J. 2021. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, 1–10. IEEE.

Liu, Y.; Kang, Y.; Zou, T.; Pu, Y.; He, Y.; Ye, X.; Ouyang, Y.; Zhang, Y.-Q.; and Yang, Q. 2024a. Vertical federated learning: Concepts, advances, and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 36(7): 3615–3634.

Liu, Y.; Xu, L.; Yuan, X.; Wang, C.; and Li, B. 2022. The right to be forgotten in federated learning: An efficient realization with rapid retraining. In *IEEE INFOCOM 2022-IEEE conference on computer communications*, 1749–1758. IEEE.

Liu, Z.; Jiang, Y.; Shen, J.; Peng, M.; Lam, K.-Y.; Yuan, X.; and Liu, X. 2024b. A survey on federated unlearning: Challenges, methods, and future directions. *ACM Computing Surveys*, 57(1): 1–38.

Maaten, L. v. d.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov): 2579–2605.

Mazzia, V.; Pedrani, A.; Caciolai, A.; Rottmann, K.; and Bernardi, D. 2024. A survey on knowledge editing of neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

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

Meerza, S. I. A.; Sadovnik, A.; and Liu, J. 2024. Confuse: Confusion-based federated unlearning with salience exploration. In *2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 427–432. IEEE.

Mitchell, E.; Lin, C.; Bosselut, A.; Finn, C.; and Manning, C. D. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A. Y.; et al. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning (NIPS Workshop)*, 7.

Nguyen, T.-H.; Vu, H.-P.; Nguyen, D. T.; Nguyen, T. M.; Doan, K. D.; and Wong, K.-S. 2024. Empirical study of federated unlearning: Efficiency and effectiveness. In *Asian Conference on Machine Learning*, 959–974. PMLR.

Pan, Z.; Wang, Z.; Li, C.; Zheng, K.; Wang, B.; Tang, X.; and Zhao, J. 2025. Federated unlearning with gradient descent and conflict mitigation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 19804–19812.

Varshney, A. K.; Vandikas, K.; and Torra, V. 2025. Unlearning Clients, Features and Samples in Vertical Federated Learning. *arXiv preprint arXiv:2501.13683*.

Vepakomma, P.; Gupta, O.; Swedish, T.; and Raskar, R. 2018. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*.

Voigt, P.; and Von dem Bussche, A. 2017. The eu general data protection regulation (gdpr). *A practical guide, 1st ed., Cham: Springer International Publishing*, 10(3152676): 10–5555.

Wang, J.; Guo, S.; Xie, X.; and Qi, H. 2022. Federated unlearning via class-discriminative pruning. In *Proceedings of the ACM web conference 2022*, 622–632.

Wang, W.; Zhang, C.; Tian, Z.; and Yu, S. 2023. Machine unlearning via representation forgetting with parameter self-sharing. *IEEE Transactions on Information Forensics and Security*, 19: 1099–1111.

Wang, Z.; Gao, X.; Wang, C.; Cheng, P.; and Chen, J. 2024. Efficient vertical federated unlearning via fast retraining. *ACM Transactions on Internet Technology*, 24(2): 1–22.

Wu, W.; Gong, Y.; Jiang, J.; Hu, C.; Zhou, X.; and Cheng, D. 2025a. Defending against Attribute Inference Attacks in Post-Training of Recommendation Systems via Unlearning. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, 2656–2669. IEEE Computer Society.

Wu, W.; Liang, H.; Tu, T.; Jiang, J.; Hu, C.; and Cheng, D. 2025b. Mimir: Data-Free Federated Unlearning Through Client-Specific Prompt Generation for Personalized Models. *IEEE Transactions on Mobile Computing*.

Wu, W.; Liang, H.; Yuan, J.; Jiang, J.; Wang, K. Y.; Hu, C.; Zhou, X.; and Cheng, D. 2025c. Zero-shot Federated Unlearning via Transforming from Data-Dependent to Personalized Model-Centric. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, 6588–6596. Main Track.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xie, Z.; Gao, Z.; Lin, Y.; Zhao, C.; Yu, X.; and Chai, Z. 2024. Adaptive Clipping and Distillation Enabled Federated Unlearning. In *2024 IEEE International Conference on Web Services (ICWS)*, 748–756. IEEE.

Yang, Q.; Liu, Y.; Cheng, Y.; Kang, Y.; Chen, T.; and Yu, H. 2020. Horizontal federated learning. In *Federated learning*, 49–67. Springer.

Yu, T.; Kumar, S.; Gupta, A.; Levine, S.; Hausman, K.; and Finn, C. 2020. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836.

Zhao, K.; Kurmanji, M.; Bărbulescu, G.-O.; Triantafillou, E.; and Triantafillou, P. 2024a. What makes unlearning hard and what to do about it. *Advances in Neural Information Processing Systems*, 37: 12293–12333.

Zhao, Z.; Liang, X.; Huang, H.; and Wang, K. 2024b. Deep federated learning hybrid optimization model based on encrypted aligned data. *Pattern Recognition*, 148: 110193.

Zhu, X.; Li, G.; and Hu, W. 2023. Heterogeneous federated knowledge graph embedding learning and unlearning. In *Proceedings of the ACM web conference 2023*, 2444–2454.