Unlearning Clients, Features and Samples in Vertical Federated Learning

Ayush K. Varshney*
Umeå University
Umeå, Sweden
ayushkv@cs.umu.se

Konstantinos Vandikas
Ericsson Research, Ericsson
Stockholm, Sweden
konstantinos.vandikas@ericsson.com

Vicenç Torra
Umeå University
Umeå, Sweden
vtorra@cs.umu.se

Abstract

Federated Learning (FL) has emerged as a prominent distributed learning paradigm that allows multiple users to collaboratively train a model without sharing their data thus preserving privacy. Within the scope of privacy preservation, information privacy regulations such as GDPR entitle users to request the removal (or unlearning) of their contribution from a service that is hosting the model. For this purpose, a server hosting an ML model must be able to unlearn certain information in cases such as copyright infringement or security issues that can make the model vulnerable or impact the performance of a service based on that model. While most unlearning approaches in FL focus on Horizontal Federated Learning (HFL), where clients share the feature space and the global model, Vertical Federated Learning (VFL) has received less attention from the research community. VFL involves clients (passive parties) sharing the sample space among them while not having access to the labels. In this paper, we explore unlearning in VFL from three perspectives: unlearning passive parties, unlearning features, and unlearning samples. To unlearn passive parties and features we introduce VFU-KD which is based on knowledge distillation (KD) while to unlearn samples, VFU-GA is introduced which is based on gradient ascent (GA). To provide evidence of approximate unlearning, we utilize Membership Inference Attack (MIA) to audit the effectiveness of our unlearning approach. Our experiments across six tabular datasets and two image datasets demonstrate that VFU-KD and VFU-GA achieve performance comparable to or better than both retraining from scratch and the benchmark R2S method in many cases, with improvements of $(0 - 2\%)$. In the remaining cases, utility scores remain comparable, with a modest utility loss ranging from $1 - 5\%$. Unlike existing methods, VFU-KD and VFU-GA require no communication between active and passive parties during unlearning. However, they do require the active party to store the previously communicated embeddings.

Keywords

Federated learning; Unlearning; Vertical federated learning; Auditing; Membership Inference Attack (MIA).

*Corresponding author.

Proceedings on Privacy Enhancing Technologies 2025(2), 39–53
© 2025 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2025-0048

1 Introduction

Machine Learning (ML) models have established themselves as the prominent artificial intelligence (AI) approach due to their ability to learn complex patterns from large-scale user data. The huge amount of data used to train these models is often sensitive in nature. To safe guard the information and privacy of the users, information privacy regulations such as GDPR, and CCPA have been proposed. These regulations allow users the right to be forgotten, i.e., to remove their data and its influence from the ML model.

Removing the influence of the data or *Unlearning* the data is a challenging task. A naive approach is to retrain the model in the absence of the data to remove. This approach can be time-consuming and assumes that the original data is available. A more attractive approach should remove the data and its influence without the need of retraining from scratch. The objective is to modify the model parameters of a ML model in such a way that the modified parameters are the same as those parameters of a model that would have been retrained from scratch with an original dataset deprived of the data to be forgotten or unlearned. However, achieving such an objective can be computationally expensive. An approach proposed by Bourtoule et al. [3] divides the dataset into shards, retraining only the shard containing the data to be removed upon receiving an unlearning request. While this method minimizes the scope of retraining, it may struggle to capture complex relationships across shards and becomes computationally expensive for frequent unlearning requests. To address such challenges, Ginart et al. [9] introduced the concept of approximate unlearning, where model parameters are adjusted to closely approximate those of a retrained model, reducing the need for full retraining. This concept has since been extended in several studies. For instance, Halimi et al. [12] propose updating the model using gradient ascent on the data to be forgotten. Wu et al. [35] leverage knowledge distillation for unlearning specific samples, while Tarun et al. [29] refine model parameters through modified fine-tuning for efficient unlearning.

The majority of existing approaches in machine unlearning operate under the assumption that the original dataset, or the specific data points to be removed, are readily available, refer [39] for more information. These methods typically rely on access to the dataset for performing the unlearning process, which involves directly manipulating or retraining the model on the modified data. This assumption simplifies the unlearning task but may not be practical or feasible in many real-world scenarios where data access is restricted due to privacy concerns, regulatory requirements, or logistical constraints.

In a distributed paradigm like Federated Learning (FL) [25], clients collaboratively train a ML model without sharing their data. They only communicate the model weights to a server, which in turn aggregate the weights from clients to create a global model. This continues for several rounds to learn complex relationships between client data. Based on the nature of the data distributed among clients FL is further classified into three types, horizontal, vertical and transfer FL [27]. In HFL, clients share the same feature space, but the samples and their distributions differ across each client. This type is suitable for a large number of heterogeneous devices in a complex network. However, its application in collaboration between companies or institutions is practically restricted due to its limited ability to handle different feature space. In VFL, clients share the same sample space but have different feature space, with only the server having access to the labels. This setup enables institutions with distinct feature spaces to collaborate on training ML models while ensuring the privacy of their raw data. Notably, data alignment is required in VFL before training. To achieve this, clients utilize private set intersection (PSI) [7] to identify the common sample space while preserving data privacy. For further details on PSI and its drawbacks, refer to Wu et al.[36]. With Federated Transfer Learning (FTL), the server leverages knowledge from one domain to another, facilitating collaborative training of ML models across different domains. FTL enables knowledge transfer and model training even when clients have different feature spaces and sample distributions.

Unlearning in FL has its own unique challenges. In each communication round in FL, the client contribution is aggregated and communicated to the rest of the clients. The server does not have access to the client data. The role of the unlearner depends on the unlearning query. For example, Wang et al. [31] highlights that when a client request to unlearn the server acts as the unlearner, and is responsible for adjusting the parameter of the global model; when a client request to unlearn a portion of their data, the client themselves are the unlearner and update their local model parameters. Jiang et al. [15] remove the contributions from the client iteratively based on its historical data, Zhu et al. [41] unlearn the model by averaging the model updates from the remaining clients while optimizing distillation loss between the original model and unlearned model, Li et al. [18] propose a gradient ascent based approach to unlearn a client weight, Wu et al. [35] utilize Knowledge Distillation (KD) for federated unlearning and many more such methods. Majority of the unlearning literature in FL has focused on HFL. In HFL, the model architecture remains the same, therefore the server can store historical updates from the clients and produce a calibrated unlearned model for approximate unlearning [20]. However, the model architecture in VFL may change depending on the type of unlearning required. As a result, existing unlearning approaches designed for HFL cannot be directly applied to VFL without significant modifications.

In VFL, each training round involves communicating embeddings from clients (passive parties) to the server (active party), making the training process communication intensive. Consequently, any unlearning approach that necessitates even a few additional training rounds between the active and passive parties is undesirable due to the communication overhead it imposes. Also, the model architecture of the active party depends on the embeddings from the passive parties i.e., removing the embeddings of a passive party reduces the size of the input layer in the active model. Here as well, the unlearner can be different based on the unlearning request.

The literature for unlearning in VFL is scarce, Deng et al. [6] propose one of the first unlearning approach in VFL, however it is restricted to logistic regression. Their approach stores the last communication round embeddings from each client, and assumes unlearning request can come only after the completion of training process, which is not realistic. The learning in FL is continuous in nature and unlearning request can come during training as well. Another approach [32] proposes a fast retraining method for VFL by storing bottom model checkpoints. These bottom model checkpoints are used to reinitialize the passive parties. Their approach proposes fast retraining method which maintains several passive party models which are used to reinitialize and retrain the passive models from scratch. Their retraining approach for unlearning in VFL requires communication between active and passive parties, which is costly. Additionally, for feature unlearning and passive party unlearning—i.e., when a passive party wishes to remove certain features from its local model, and when a passive party needs to be unlearned from the global model—this often requires reducing model parameters, which in turn requires retraining from scratch in the existing literature. Notably, none of the existing approaches in the VFL unlearning literature address the challenges of feature, sample, and passive party unlearning simultaneously.

Overall, we find that for passive party unlearning and feature unlearning, approaches in VFL should be able to deal with the reduction in model size. It should not involve any communication between the active and passive parties. Considering these challenges in mind, we propose a knowledge distillation based unlearning approach for VFL which we call Vertical Federated Unlearning with Knowledge Distillation (VFU-KD). The advantage of unlearning with KD, is its ability to handle model compression well and it does not require communication between active and passive parties. We also extend VFU-KD to propose the first feature unlearning approach for a passive party in a VFL. For sample unlearning in VFL, unlearning with KD is an expensive approach as sample unlearning does not require model compression. On the other hand, gradient ascent [11] is a popular approach to unlearn samples in centralized machine learning. Considering this in mind, we propose a gradient ascent based unlearning for VFL called Vertical Federated Unlearning with Gradient Ascent (VFU-GA) which maximize the loss on the forget set and fine tune the model for few epochs on the remaining data for approximate unlearning.

Auditing the unlearning algorithm is also crucial to verify that the unlearning has been done. In the literature, the majority of the approaches use the drop in accuracy of a backdoor attack [2] as the sign of unlearning. However, in case of unlearning in VFL, backdoor attacks can not be used to audit unlearning as once the passive party embeddings are removed, it is not possible to place the backdoor in the active model. The data poisoning attack proposed in Deng et al. [6] considers the availability of labels to the passive party[1] which violates the fundamental assumptions in the VFL setting. In our work, we propose a membership inference attack (MIA) [26], which does not violate the VFL constraints and can be used to audit unlearning in VFL.

In summary, we make the following contributions.

[1]https://github.com/dateaaalive/vfl/blob/main/data_poison_attack.py

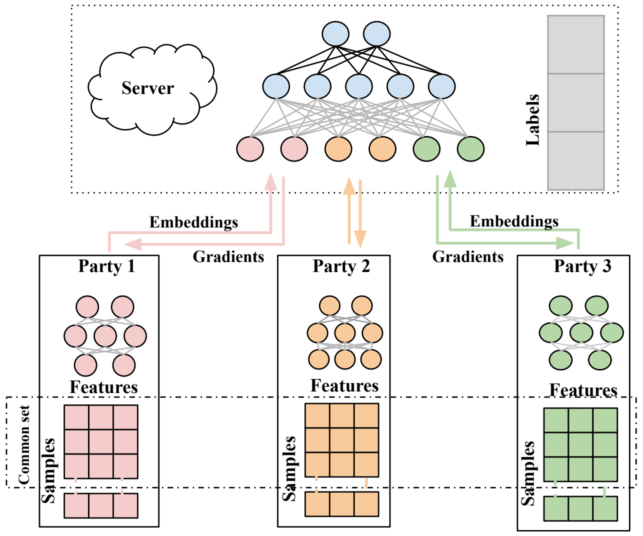


Figure 1: Vertical federated learning framework

(1) A novel vertical federated unlearning framework with knowledge distillation to unlearn a passive party.
(2) A novel feature unlearning framework for a passive party in VFL.
(3) A novel sample unlearning framework using gradient ascent in VFL.
(4) A membership inference attack to audit the unlearning in VFL.
(5) Empirical analysis on tabular as well as image datasets.

The rest of the paper is organized as follows. Section 2 provides the necessary background. Section 3 describes our proposed frameworks. Section 4 gives the experimental analysis. The paper finishes with conclusion and some future work in Section 5.

2 Background

2.1 Vertical federated learning

The demand for VFL has grown rapidly in recent years [22]. VFL is a distributed learning paradigm that allows organizations with distinct feature spaces to collaboratively train ML model while safeguarding the privacy of their raw data. This approach is particularly valuable in scenarios where data privacy is critical, such as in healthcare [38], finance [40], and cross-enterprise collaborations [23]. In these domains, institutions often possess complementary datasets but are constrained from sharing them due to regulatory requirements or competitive concerns. In a typical VFL setup, there are N passive parties (i.e., clients) that own data but lack access to the labels, and a single active party, typically the server or a trusted third party, that holds the labels as shown in Fig. 1. This ensures that the learning process can leverage distributed data sources while maintaining strict privacy guarantees. Organizations and institutions with limited and fragmented datasets constantly seek data partners to collaboratively train ML models to maximize data utilization [19]. VFL requires all participants to share sample space but allows for different feature spaces. This essentially suggests less number of participants in VFL (Wei et al. [34] suggests $N < 5$), with two-party VFL as the most common setting. In VFL, the first step is to perform private set intersection [24] to identify common sample IDs among participating parties while preserving privacy. Each passive party can have a local model, referred to as a passive model. Based on the common set from private set intersection, passive parties do a forward pass on their local models. The computed embeddings are communicated to the active party, which concatenates the embeddings from the passive parties and then performs a forward pass on its model, referred to as the active model. The active party computes the loss and backpropagates the gradients to the active model and to the respective embeddings. The active party then communicates the respective gradients to each passive party. Upon receiving the gradients, passive parties update their local models. It is important to highlight that this process continues for each batch and requires several communication rounds between the active and passive parties for one epoch of training data. Hence, training in VFL is a communication-intensive algorithm [34].

2.2 Knowledge distillation

KD approaches has been used in the literature for their ability to transfer knowledge from a bigger teacher model to a smaller student model [13]. The simple idea of KD is that student model tries to mimic the output probabilities of the teacher model. Usually, student model learns from its own loss and its divergence from the teacher's logits. For a labeled dataset with $\hat{y}$ as true labels, the loss function for the student model can be given as follows:

$$\mathcal{L} = (1-\alpha)L_{pred}(\hat{y}, \hat{y}_{student}) + KL\_DIV(\hat{y}_{student}, \hat{y}_{teacher}) \quad (1)$$

here α is used to manage the trade-off between distillation loss and prediction loss; $\hat{y}_{student}$, $\hat{y}_{teacher}$ are the logits of the student and teacher models respectively; and $KL\_DIV()$ is the KL divergence between them. Please refer [10] for a discussion of recent advancements in the area of KD.

2.3 Gradient ascent

Gradient ascent is the counterpart of gradient descent which is the typical optimization approach used to train a machine learning model. In gradient descent, the objective is to minimize the loss function for a given set of samples, whereas in gradient ascent (GA), the goal is to maximize the loss function for those samples. This approach is particularly useful for unlearning, as it allows for the approximate removal of specific samples by adjusting the model weights in the direction that maximizes the loss on the target samples [29], [30]. Let θ represent the model weight, η be the learning rate and L be the loss function, then Gradient Ascent (GA) iteratively updates the model weights in the following manner.

$$\theta_{t+1} = \theta_t + \eta \frac{\delta L}{\delta \theta_t} \quad (2)$$

2.4 Membership inference attack

MIA enables adversaries to determine whether a particular record was part of the training set in a ML model. MIA leverages the memorization of ML models i.e, ML models behave differently on the data seen during training. Consequently by analyzing the model's response on various inputs, an attacker can infer the presence or absence of specific data during training.

3 Proposed Work

In this section, we present the details of the proposed unlearning framework for passive party unlearning, sample unlearning and feature unlearning in VFL. We also provide the details of MIA used to audit unlearning in our framework. Unlearning is a crucial capability for addressing a variety of issues, including privacy compliance, security, and adaptability. Existing approaches either violate VFL constraints or are expensive in terms of communication for passive party unlearning. None of the approaches (to the best of our knowledge) focuses on feature unlearning in VFL.

Algorithm 1 shows the generic VFL framework with K passive parties, and K^{th} party being the active party as well. Let $\mathcal{G}()$ be a function which takes model parameters θ, and minibatch x as input and returns the embeddings from the model.

Passive parties have their local models $\theta_1, \theta_2, ..., \theta_K$ and Θ_K is the active model. For each batch, passive parties do a forward pass and communicate their embeddings to the server. Server in turn trains its local model and forward the gradients with respect to each embedding ($\nabla^t_k L = \frac{\delta L}{\delta \theta^t_k} = \frac{\delta L}{\delta \Theta^t_K} \times \frac{\delta \Theta^t_K}{\delta \theta^t_k}$, L being the loss function) back to their respective passive party. Algorithm 1 requires parameters η_1, η_2 which needs to be calibrated to have a successful learning. However this can be eliminated if we assume the loss L to be twice differential and strictly convex, then the parameter update can be written as:

$$\Theta^{t+1}_K = \Theta^t_K - \mathcal{H}^{-1}_{\Theta^t_K} \frac{\delta L}{\delta \Theta^t_K} \tag{3}$$

And for passive parties:

$$\theta^{t+1}_k = \theta^t_k - \mathcal{H}^{-1}_{\theta^t_k} \nabla^t_k L \tag{4}$$

where $\mathcal{H}$ is the hessian matrix for each model. [33] suggests computing and storing comes at an additional computational cost of $O(np^2 + p^3)$ and $O(p^2)$ respectively. We have considered both the approaches i.e., using η_1, η_2 and $\mathcal{H}^{-1}$ for the respective models.

To unlearn a passive party, the active party must remove the contribution of the target passive party from all of its historical embeddings, i.e., the active party updates $H = H \setminus H_u$ in all the training rounds, H_u being the target passive party. In our work, we use the KD approach as the unlearning mechanism as it can deal with model compression, and since the active party already has the embeddings from the previous rounds, the active party does not need to have any communication between active and passive parties.

In our approach, we first randomly initialize a new student model to eliminate any previous information, and assign the old active model as the teacher model. The active party updates its student model based on the historical embeddings from the rest of the clients. The loss for the student model is the combination of prediction loss and the distillation loss.

$$L = \alpha * L_{distil} + (1 - \alpha) * L_{pred} \tag{5}$$

The parameter α balances the trade-off between prediction loss (L_{pred}) and distillation loss (L_{distil}). In our case, we have considered the $KL\_DIV()$ between as the distillation loss. The output probabilities of the teacher model is used to guide the training of the student model, but our approach is not restricted to it, other distillation functions such as in [10] can also be used. The training of student model considers stored embedding for each batch and computes overall loss as defined in eq. 5. This process continues till the student model converges or till the given number of epochs.

Algorithm 1: Vertical Federated Learning Algorithm

```
Input: Passive parties $\theta_1, \theta_2, ..., \theta_K$, Active party $\Theta_K$,
       learning rates $\eta_1, \eta_2$
Output: Trained model weights $\theta_1, \theta_2, ..., \theta_K$ and $\Theta_K$
1  Randomly initialize $\theta_1, \theta_2, ..., \theta_K$ and $\Theta_K$.
2  for $t = 1, 2, \ldots, T$ do
3      Randomly sample a minibatch $x \in \mathcal{D}$
4      for each party $k = 1, 2, ..., K$ in parallel do
5          Party $k$ does a forward pass and computes
             embeddings $H_k = \mathcal{G}(\theta^t_k, x)$
6          Party $k$ sends $H_k$ to the Active party
7      Active party computes $H^t = concat(H_1, H_2, ..., H_k)$
8      Active party stores $H^t$
9      Compute prediction $\hat{y} = \Theta^t_K(H^t)$
10     Compute loss $L = LossFunction(\hat{y}, y_{true})$
11     Active party updates $\Theta^{t+1}_K = \Theta^t_K - \eta_1 \frac{\delta L}{\delta \Theta^t_K}$
12     Active party computes $\frac{\delta L}{H_k}$ and sends it to the respective
         passive parties
13     for each party $k = 1, 2, ..., K$ in parallel do
14         Party $k$ computes $\nabla^t_k L = \frac{\delta L}{\delta \theta^t_k} = \frac{\delta L}{\delta \Theta^t_K} \times \frac{\delta \Theta^t_K}{\delta \theta^t_k}$
15         Party $k$ updates $\theta^{t+1}_k = \theta^t_k - \eta_2 \nabla^t_k L$
```

Algorithm 2: Passive Party Unlearning in VFL

```
Input: Target passive party $\theta_u$, Active party $\Theta_K$, learning
       rate $\eta_1$, distillation rate $\alpha$, current epoch $ep$
Output: Unlearned Active party model $\Theta_K$, Updated
        historical embeddings
1  Randomly initialize $\Theta_{student}$ with input size of $H \setminus H_u$
2  Assign $\Theta_K$ as teacher model $\Theta_{teacher}$
3  for $t = 1, 2, \ldots, ep$ do
4      Active party reads $H^t$
5      Compute teacher prediction $\hat{y}_{teacher} = \Theta_{teacher}(H^t)$
6      Update $H^t = H^t \setminus H_u$    // $H_u$ is the target party
         $\theta_u$ embeddings
7      Active party removes the old $H^t$ and stores the updated
         $H^t$
8      Compute prediction $\hat{y}_{student} = \Theta^t_{student}(H^t)$
9      Compute prediction loss
         $L_{pred} = LossFunction(\hat{y}_{student}, y_{true})$
10     Compute distillation loss
         $L_{distil} = KL\_DIV(\hat{y}_{student}, \hat{y}_{teacher})$
11     Overall loss $L = \alpha * L_{distil} + (1 - \alpha) * L_{pred}$
12     Active party updates $\Theta^{t+1}_{student} = \Theta^t_{student} - \eta_1 \frac{\delta L}{\delta \Theta^t_{student}}$
```

Algorithm 2 shows the formal algorithm for unlearning a passive party in VFL setting. Here as well, if the loss is twice differential and strictly convex the model update can be written as:

$$\Theta^{t+1}_{student} = \Theta^{t}_{student} - \mathcal{H}^{-1}_{\Theta_{student}} \frac{\delta L}{\delta \Theta^{t}_{student}} \tag{6}$$

Algorithm 3: Feature Unlearning for a Passive Party in VFL

```
Input: Target passive party $\theta_u$, target features $f_u \subset f$,
       learning rate $\eta_2$, current epoch $ep$
Output: Unlearned passive party model $\theta_u$
1 Randomly initialize $\theta_{student}$ with input size of $f \setminus f_u$   // $f$
    -> feature space of $\theta_u$
2 Assign $\theta_u$ as teacher model $\theta_{teacher}$
3 for $t = 1, 2, \ldots, ep$ do
4     Randomly sample a minibatch $x \in \mathcal{D}$
5     Compute teacher embedding
        $emb_{teacher} = H^t_u = \mathcal{G}(\theta_{teacher}, x)$
6     Compute student embeddings
        $emb_{student} = \mathcal{G}(\theta^t_{student}, x \setminus x_u)$
7     Compute loss $L = KL\_DIV(emb_{student}, emb_{teacher})$
8     Passive party updates $\theta^{t+1}_{student} = \theta^t_{student} - \eta_2 \frac{\delta L}{\delta \theta^t_{student}}$
```

With the ever-changing privacy regulations all over the world, passive parties must have the ability to remove/unlearn the influence of controversial features e.g, sensitive features such as gender, ethnicity are unlikely to be used in training the ML model anymore. In case of VFL, since passive parties do not have access to the true labels (only their embeddings, say *emb*), computing prediction loss for the student model is not possible without any communication with the active party. Our goal is to minimize or not have communication between active and passive parties for unlearning. Hence, the distillation loss is the overall loss ($L = KL\_DIV(emb_{student}, emb_{teacher})$) for the student model in this case. Algorithm 3 presents the formal feature unlearning algorithm for VFL. Here as well, we randomly initialize a student model with the input size of new feature space. In each training round, the student model updates its parameter with the $KL\_DIV()$ from its teacher model. And its hessian variation can be written as:

$$\theta^{t+1}_{student} = \theta^{t}_{student} - \mathcal{H}^{-1}_{\theta_{student}} \frac{\delta L}{\delta \theta^{t}_{student}} \tag{7}$$

Algorithm 3 does not require any additional storage for embeddings at the passive party as the teacher and student models can compute the embeddings from the randomly sampled minibatch.

The model parameters in Algorithm 2 and 3 are vectors in high-dimensional model space. Consider that the teacher model is in $\mathbb{R}^n$ i.e., $\Theta^t_{teacher} \in \mathbb{R}^n$ and the student model is in $\mathbb{R}^m$ i.e., $\Theta^t_{student} \in \mathbb{R}^m, m < n$. The embedding $H^t \setminus H_u$ lies in the lower dimension manifold of H^t. Consequently, the posterior $p(\Theta_{teacher}|H)$ is formed with broader set of features, potentially leading to a more complex model. Similarly, the posterior $p(\Theta_{student}|H \setminus H_u)$ is formed with a reduced set of features, meaning $\Theta_{student}$ would lose some information compared to the $\Theta_{teacher}$ model. With each training epoch, the $KL\_DIV((\hat{y}_{student}|H \setminus H_u)||(\hat{y}_{teacher}|H))$ increases due to the loss of information in $\Theta_{student}$ until the model converges. The $KL\_DIV((\hat{y}_{student}|H \setminus H_u)||(\hat{y}_{teacher}|H)) > 0$ can be lower bounded by some δ, which further can be used to measure the degree of unlearning. Similarly, for feature unlearning, the features $f \setminus f_u$ are in the lower dimension manifold of f, and $(KL\_DIV(emb_{student}|f \setminus f_u)||emb_{teacher}|f)$ can be lower bounded by some δ_f which can be used to measure the degree of unlearning.

Algorithm 4: Sample Unlearning in VFL

```
Input: Target batch $id$ corresponding tosamples to unlearn,
       Active party $\Theta_K$, learning rate $\eta_1$, unlearning rate $\lambda$,
       unlearning epochs $u_{ep}$, current epoch $ep$
Output: Unlearned Active party model $\Theta_K$
1  Assign $\Theta_K$ as teacher model $\Theta_{teacher}$
2  for $t = ep - u_{ep}, \ldots, ep$ do
3      Active party reads $H$
4      Find $H_u = H_{id}$          // $H_u$ is the embeddings
         corresponding to the target samples
5      Update $H = H \setminus H_u$
6      Compute prediction $\hat{y}_{retain} = \Theta^t_K(H)$
7      Compute prediction $\hat{y}_{target} = \Theta^t_K(H_u)$
8      Compute loss $L_{retain} = LossFunction(\hat{y}_{retain}, y_{true})$
9      Compute loss $L_{target} = LossFunction(\hat{y}_{target}, y_{true})$
10     Active party updates $\Theta^{t+1}_K = \Theta^t_K - \eta_1 \frac{\delta L_{retain}}{\delta \Theta^t_K} + \lambda \frac{\delta L_{target}}{\delta \Theta^t_K}$
```

Unlearning samples in VFL does not require model compression, making knowledge distillation KD a computationally expensive approach for this task, as it involves retraining the model. To make sample unlearning more efficient, we propose using gradient ascent, which significantly reduces training time. Our approach approximately unlearns samples by maximizing the model's loss on the target set (the samples to be unlearned) while minimizing the loss on the remaining dataset (the retain set) over a specified number of epochs. Algorithm 4 presents the formal algorithm for sample unlearning in VFL. Here, the algorithm requires batch ids corresponding to the samples in target set and the number of unlearning epochs (u_{ep}). The unlearning epochs determines how many retraining rounds will be performed. The active party computes the loss on both the target and retain sets, updating its model parameters to maximize the loss on the target set and minimize the loss on the retain set i.e., if L_{retain} and L_{target} is the loss on retain set and target set then the model update can be written as:

$$\Theta^{t+1}_K = \Theta^t_K - \eta_1 \frac{\delta L_{retain}}{\delta \Theta^t_K} + \lambda \frac{\delta L_{target}}{\delta \Theta^t_K} \tag{8}$$

When the target set is absent then, the updates simplifies to:

$$\Theta^{t+1}_K = \Theta^t_K - \eta_1 \frac{\delta L_{retain}}{\delta \Theta^t_K} \tag{9}$$

The choice of unlearning rate λ plays a crucial role in determining the speed of convergence and stability. A poor choice of λ may result in the model parameters getting stuck in local optimum. In comparison with fine-tuning (eq. 9), the model accelerates with

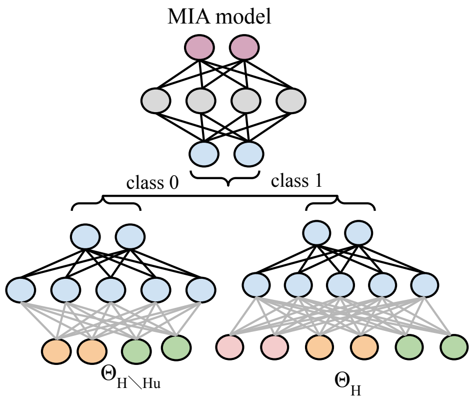


Figure 2: MIA attack model.

$\lambda \frac{\delta L_{target}}{\Theta^t_K}$ to quickly unlearn the updates from target set with Algorithm 4. The choice of λ and u_{ep} can further be used to determine the rate of unlearning.

3.1 Auditing the unlearning

Auditing the unlearning process in VFL is especially crucial when institutions with commercial interests are involved. It ensures that proprietary and sensitive information shared among parties are thoroughly removed upon request, thus complying with information privacy regulations such as GDPR. Effective auditing maintains the trust of all participating institutions by verifying that data removal processes are complete and accurate, thereby preventing any residual influence of deleted data on the model's predictions.

In the literature of unlearning in FL data poisoning [6] and backdoor [35] attacks are the most common ways to verify unlearning. Specifically, data poisoning attacks involve injecting malicious data into the training process to manipulate the model's behavior, thereby challenging the unlearning system's ability to completely remove the influence of such tampered data [6]. On the other hand, backdoor attacks involve embedding hidden triggers within the model, which cause it to behave maliciously when encountering specific inputs, thus providing a rigorous test of whether unlearning mechanisms can entirely eliminate such hidden backdoors [35]. However, we argue that auditing unlearning with data poisoning attack and backdoor attacks is not suitable in VFL setup. For backdoor attacks, once the embeddings of the target party are unlearned from the active model, the target party can not place the backdoor (for verification) in the new active model. A similar reasoning continues for data poisoning attack along with the absence of true labels in passive parties, thus data poisoning attack is also unsuitable to audit VFL. In our case, we have considered MIA (refer section 2.4) to audit unlearning in VFL.

For MIA, we have a binary classification model which is trained on the output probabilities of the active model in the presence (class 1) and absence (class 0) of the target party's embeddings for few epochs (see Fig. 2). Once the model is trained, its inference can be used to audit the unlearning of the target party, i.e., whether the target party participated in training the active model or not. Similarly, for auditing sample unlearning, the MIA model is trained with the output probabilities in the presence (class 1) and absence (class 0) of the samples to unlearn.

4 Experimental analysis

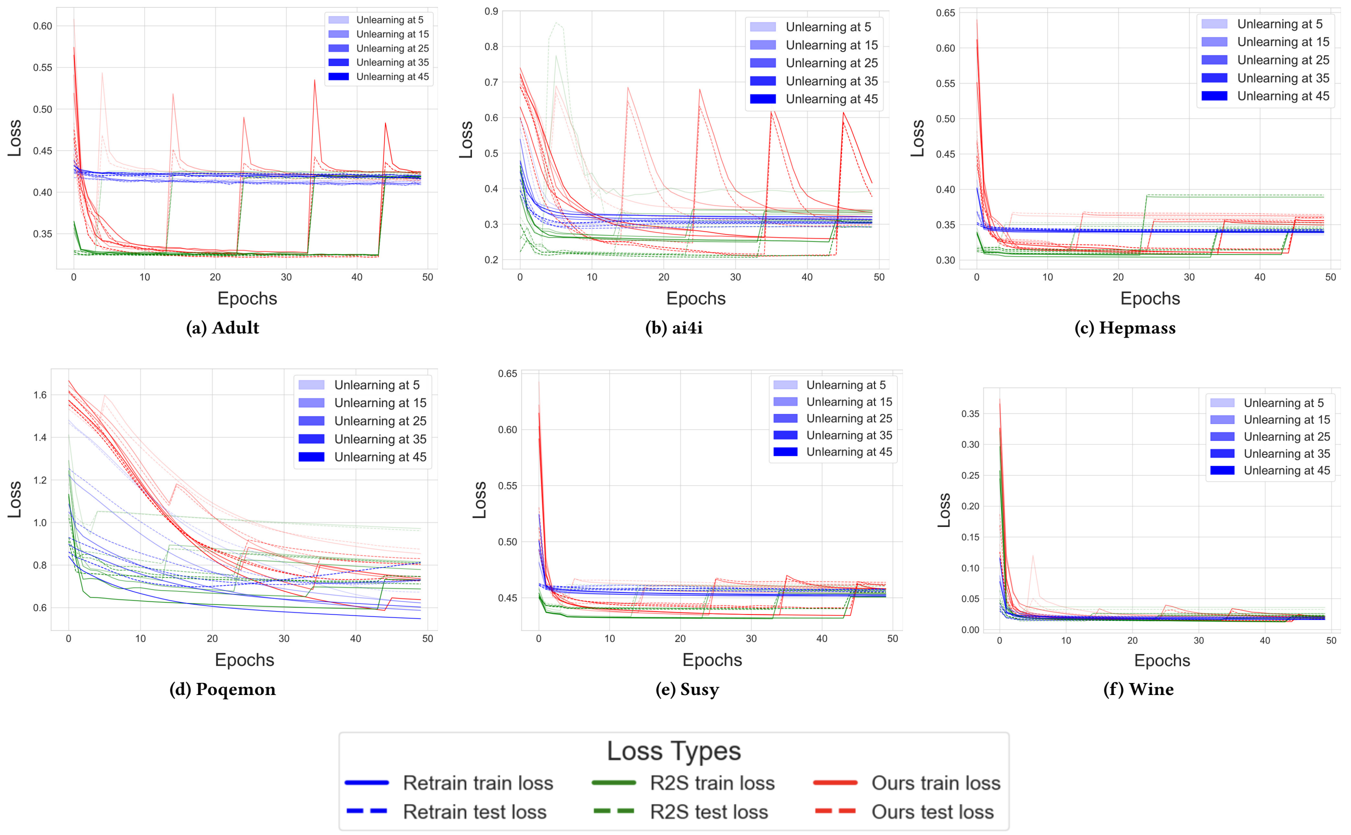
In this section, we present the experimental analysis of our proposed unlearning framework. As discussed in Section 2.1, the most common VFL setting typically involves two parties, with a maximum of four parties. For this paper, we consider a three-party VFL setup consisting of *clientA*, *clientB*, and *clientC*, collaborating to train a joint VFL model. The training process spans 50 epochs, with *clientA* having the flexibility to request unlearning at any point during the training process. Since, communication in VFL is communication-intensive, using a larger batch size is preferred to optimize efficiency [8]. Accordingly, we set the batch size to 512 in our experiments. The learning rate is set to 10^{-2} for tabular data and 10^{-3} for image data for both active and passive parties. Additionally, the distillation parameter which controls the trade-off between actual loss and distillation loss is set to 0.3. To demonstrate the effectiveness of our approach in unlearning at any stage during training, we conduct experiments at various epochs: [$5^{th}, 15^{th}, 25^{th}, 35^{th}, 45^{th}$]. After confirming the effectiveness and feasibility of our unlearning method, we fix the unlearning epoch at the 25^{th} epoch for further evaluation. Each experiment is then repeated three times to account for variability and capture uncertainty in the results.

In our experiments, we have considered 6 tabular datasets, namely Adult, ai4i, hepmass, susy, and wine dataset from UCI repository [16], poqemon dataset [1] and 2 image dataset CIFAR10 [17] and STL10 dataset from [5]. For tabular datasets, passive models have a single hidden layer models with 8 hidden neurons, and active model is also a single hidden layer model with 32 neurons followed by an output layer. The features are distributed equally among passive parties. For example, wine dataset has 12 features, *clientA* has first 4, *clientB* has next 4 and *clientC* has last 4 features. For image datasets, clients have resnet-18 model as the passive model, and the active model is a single hidden layer model with 512 neurons followed by an output layer. The number of neurons in output layer for active model depends on the output classes of the datasets, e.g., 2 for adult, 5 for poqemon, 10 for CIFAR10.

It is important to emphasize that these experimental setups are not optimized for peak performance. Our primary objective is to showcase the effects of unlearning on both tabular and image datasets. The parameters were selected arbitrarily and are not finetuned for optimal results on each dataset. Fine-tuning the experimental configurations for optimal results on each dataset is out-of-scope for this work.

4.1 Passive party unlearning

We compare our approach with the gold standard i.e., a retrained model from scratch and a benchmark R2S fast retraining unlearning approach [32]. For passive party unlearning, the R2S method is equivalent to retraining from scratch with smarter optimizer. Specifically, the R2S method switches between RAdam and SGD with momentum based on the training epoch and the predefined threshold. In our approach, we train the model using the RAdam optimizer [21]. Fig. 3 shows the training and test loss throughout the learning



(a) Adult **(b) ai4i** **(c) Hepmass**

(d) Poqemon **(e) Susy** **(f) Wine**

Figure 3: The training and test loss of VFU-KD compared to the retrained model from scratch and R2S method.

Epoch	Adult			ai4i			Hepmass			Poqemon			Susy			Wine		
	RfS	R2S	Ours	RfS	R2S	Ours	RfS	R2S	Ours	RfS	R2S	Ours	RfS	R2S	Ours	RfS	R2S	Ours
5th Ep.	0.65	0.62	**0.65**	0.87	0.87	**0.88**	0.83	0.82	0.82	0.94	0.86	0.88	0.78	0.56	**0.78**	0.98	0.96	0.97
15th Ep.	0.65	0.63	0.64	0.88	0.89	**0.89**	0.83	0.83	0.82	0.94	0.92	0.91	0.78	0.78	**0.78**	0.98	0.97	**0.98**
25th Ep.	0.66	0.63	**0.66**	0.89	0.89	**0.89**	0.83	0.81	**0.83**	0.94	0.91	0.92	0.78	0.78	**0.78**	0.98	0.97	**0.98**
35th Ep.	0.64	0.64	0.63	0.87	0.88	**0.89**	0.83	0.83	**0.83**	0.94	0.93	0.92	0.78	0.78	**0.78**	0.98	0.97	**0.98**
45th Ep.	0.64	0.64	0.63	0.87	0.87	0.83	0.83	0.83	0.82	0.94	0.93	0.93	0.78	0.78	**0.78**	0.98	0.97	**0.98**

Table 1: The AUC score comparison of VFU-KD (Ours) with the retrained-from-scratch (RfS) model and the R2S method.

Epoch	Adult			ai4i			Hepmass			Poqemon			Susy			Wine		
	RfS	R2S	Ours	RfS	R2S	Ours	RfS	R2S	Ours	RfS	R2S	Ours	RfS	R2S	Ours	RfS	R2S	Ours
5th Ep.	0.77	0.75	**0.77**	0.88	0.90	0.87	0.83	0.82	0.82	0.75	0.56	0.54	0.78	0.43	**0.78**	0.99	0.98	0.98
15th Ep.	0.77	0.76	**0.77**	0.88	0.89	0.87	0.83	0.83	0.82	0.76	0.67	0.70	0.78	0.78	**0.78**	0.99	0.99	**0.99**
25th Ep.	0.77	0.75	**0.77**	0.88	0.88	**0.88**	0.83	0.81	0.82	0.77	0.60	0.70	0.78	0.78	**0.78**	0.99	0.98	**0.99**
35th Ep.	0.76	0.76	**0.76**	0.88	0.89	**0.89**	0.83	0.83	0.82	0.77	0.72	0.67	0.79	0.78	0.78	0.99	0.99	**0.99**
45th Ep.	0.77	0.76	0.76	0.88	0.88	0.87	0.83	0.83	0.82	0.77	0.74	0.72	0.79	0.79	0.78	0.99	0.99	**0.99**

Table 2: The F1 score comparison of VFU-KD (Ours) with the retrained-from-scratch (RfS) model and the R2S method.

process for the tabular datasets at [$5^{th}, 15^{th}, 25^{th}, 35^{th}, 45^{th}$] epochs (Ep.). The shading indicates the epoch at which unlearning occurs: lighter shades represent earlier unlearning epochs (e.g., unlearning at the 5^{th} Ep.), while darker shades represent later epochs (e.g.,

Dataset	5th Ep.	15th Ep.	25th Ep.	35th Ep.	45th Ep.
Adult	33.4	98.6	164.3	229.2	295.0
ai4i	5.4	16.3	27.2	38.0	48.9
Hepmass	12.6	37.8	63.0	88.2	11.3
Poqemon	7.3	22	36.7	51.4	66.1
Susy	27.2	81.6	136	190.4	244.8
Wine	3.5	10.6	17.6	24.7	31.8

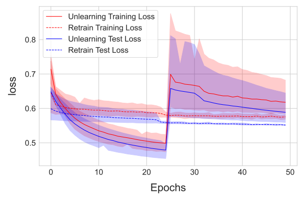
Table 3: The table presents the additional communication cost in gigabytes (GB) for unlearning with R2S method against VFU-KD.

Dataset	RfS		R2S		Ours	
	AUC	F1	AUC	F1	AUC	F1
CIFAR10	0.92	0.70	0.68	0.29	**0.92**	**0.70**
STL10	0.89	0.49	0.91	0.52	**0.89**	**0.49**

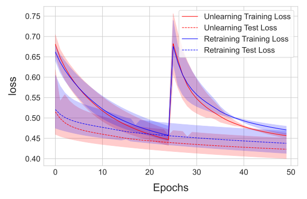
Table 4: The AUC and F1 score comparison of VFU-KD (Ours) with the retrained-from-scratch (RfS) model and the R2S method.

unlearning at the 45^{th} Ep.). Results for retraining from scratch are depicted with blue lines, the R2S method is shown with green lines, and our approach is represented with red lines. It can be clearly seen from Fig. 3 that, after unlearning at each unlearning epoch, there is a spike in the loss values for all the datasets. The spike can be attributed to the distillation process inherent in the unlearning procedure. However, the spike is not significant and the loss curve is comparable to the benchmark models, i.e., retrained model and R2S model, in all the cases. The results in Table 1 and Table 2 indicate that, for VFU-KD, the utility score (area under the curve (AUC score) and F1 score) after unlearning at the 50^{th} epoch is either similar to or better than that of retraining from scratch (abbreviated as RfS, due to limited space) and the R2S method in many cases (highlighted in bold). In the remaining cases, the utility scores are comparable, with a utility loss ranging between 1-5%. The observed improvement in loss values and utility scores may be attributed to the negative impact of *clientA* on the training of the active model. This explanation is further supported by the observation that the retrained model achieves better scores than the model trained with all three clients. Notably, our approach is not limited to the use of the RAdam optimizer; it can be seamlessly integrated with any advanced optimizer to further enhance performance. However, for the purpose of benchmarking against state-of-the-art (SOTA) methods, we specifically compare our approach using the RAdam optimizer to a retraining strategy based on the R2S optimizer. These benchmark include extensive communication with the clients. Table 3 shows significant additional communication cost required while unlearning, specially at the later epochs. This is also under an assumption that passive parties are willing to collaborate to train a new model and this also exposes the unlearn party for further privacy attacks [31].

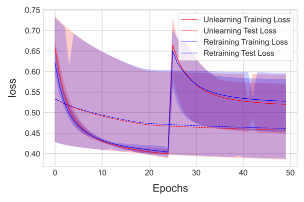
Fig. 4 shows the training and test loss when the parameters are updated with H^{-1} (refer eq. 3). The results show that the model



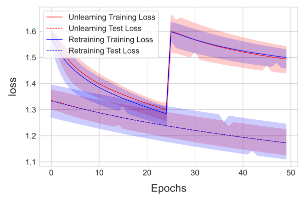
(a) Adult



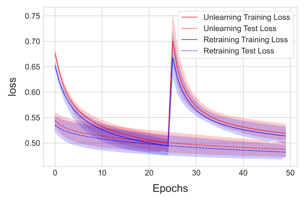
(b) ai4i



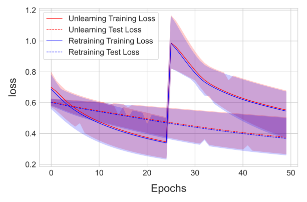
(c) Hepmass



(d) Poqemon

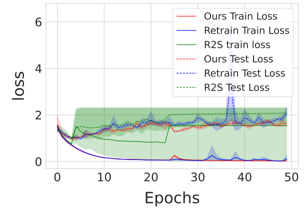


(e) Susy

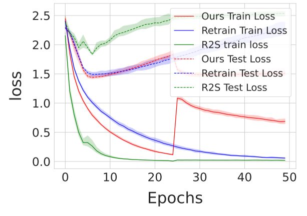


(f) Wine

Figure 4: The training (red) and test loss (blue) of VFU-KD (solid lines) with $\mathcal{H}^{-1}$ compared to the retrained model from scratch (dotted lines).



(a) CIFAR10-loss



(b) STL10-loss

Figure 5: The loss curves of VFU-KD compared to the retrained model from scratch and R2S method.

eventually has similar loss values to the benchmark model in most cases but has very sharp increase in the loss values after unlearning. From Fig. 3 and 4, we find that the results are better for unlearning with learning rates than with $\mathcal{H}^{-1}$. Further experiments with $\mathcal{H}^{-1}$ are available in supplementary material.

For comparative analysis on CIFAR10 and STL10 datasets, we argue that passive parties typically do not have incomplete images as shown in [32]. We distribute the channels of image to each client i.e., *clientA* has red channel of the image, *clientB* has green and *clientC* has blue channel of the image. Fig. 5 depicts the loss

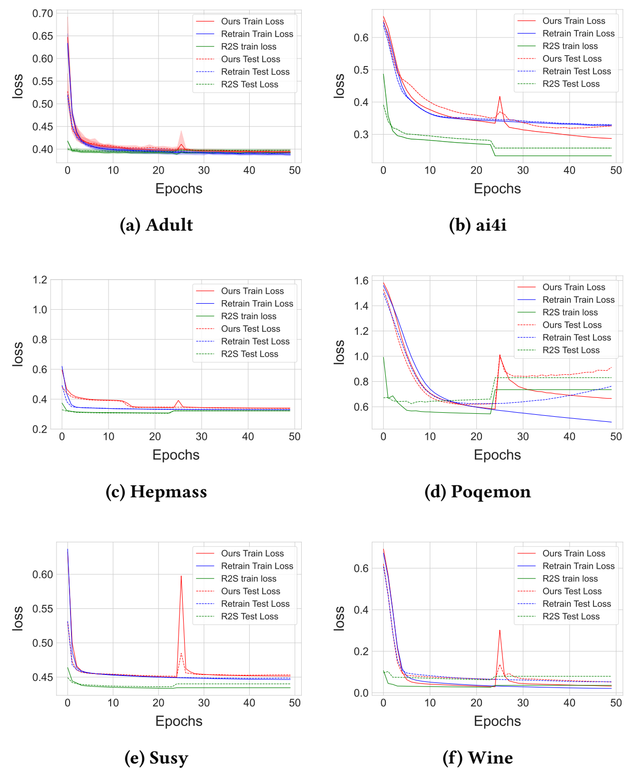


Figure 6: The training and test loss of VFU-KD for most important feature, compared to the retrained model from scratch and R2S method.

Dataset	RfS		R2S		Ours	
	AUC	F1	AUC	F1	AUC	F1
Adult	0.82	0.82	0.82	0.82	**0.82**	**0.82**
ai4i	0.86	0.87	0.87	0.90	0.86	**0.96**
Hepmass	0.84	0.84	0.85	0.85	0.84	0.84
Poqemon	0.92	0.78	0.89	0.70	0.89	0.71
Susy	0.79	0.79	0.79	0.79	**0.79**	**0.79**
Wine	0.99	0.99	0.99	0.99	**0.99**	**0.99**

Table 5: The AUC and F1 score comparison of VFU-KD (Ours) with the retrained-from-scratch (RfS) model and the R2S method for unlearning the most important feature.

curves when unlearning occurs at the 25th epoch. Small spikes in the loss curves, attributed to the distillation process, are observed but recover within a few epochs across all cases. Table 4 provides a comparison of AUC and F1 scores, demonstrating that our method achieves results equivalent to those obtained through retraining from scratch. From the results of Fig. 3, and 5, we can say that our approach, VFU-KD, has benchmark comparable losses and utility scores across 6 tabular and 2 image datasets.

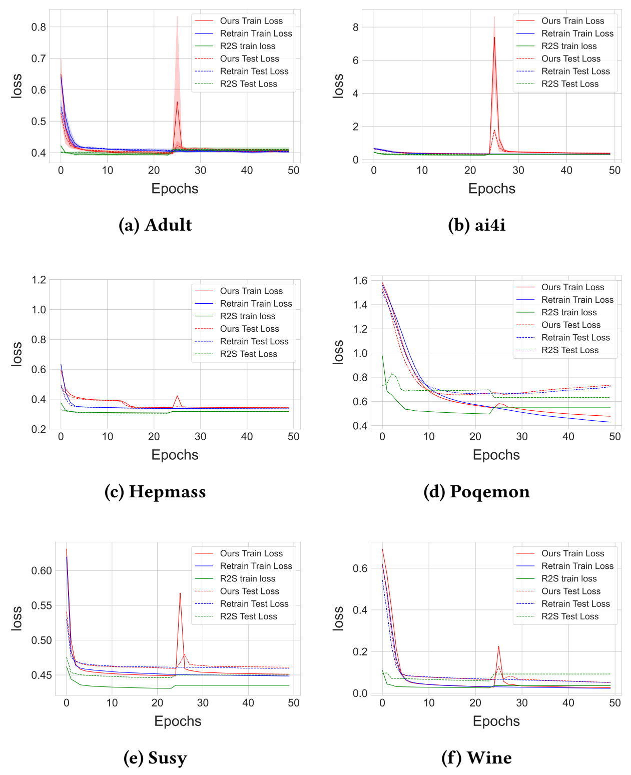


Figure 7: The training and test loss of VFU-KD for least important feature, compared to the retrained model from scratch and R2S method.

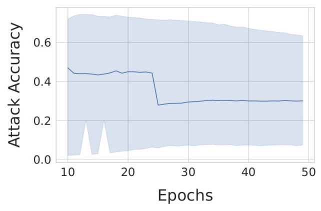
Dataset	RfS		R2S		Ours	
	AUC	F1	AUC	F1	AUC	F1
Adult	0.82	0.82	0.80	0.82	0.81	**0.82**
ai4i	0.86	0.85	0.87	0.85	**0.86**	0.81
Hepmass	0.84	0.84	0.85	0.85	0.84	0.84
Poqemon	0.92	0.79	0.92	0.79	**0.92**	**0.79**
Susy	0.78	0.79	0.79	0.79	**0.78**	**0.79**
Wine	0.99	0.99	0.99	0.99	**0.99**	**0.99**

Table 6: The AUC and F1 score comparison of VFU-KD (Ours) with the retrained-from-scratch (RfS) model and the R2S method for unlearning the least important feature.

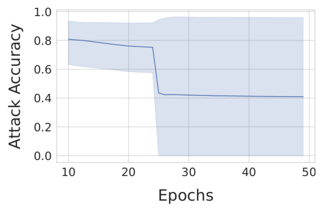
4.2 Feature unlearning

Now that we have shown the feasibility and effectiveness of our approach specially at the later epochs. We fix the unlearning epoch at $25th$ for feature and sample unlearning. For feature unlearning, we have removed the most important and least important features from tabular datasets. The importance of the feature is computed with feature ablation[2] (feature importance for all the tabular dataset is available in supplementary material). Fig. 6 shows the training

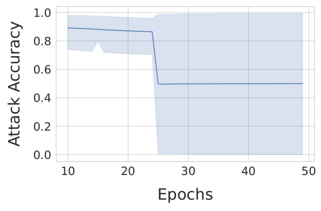
[2]https://captum.ai/api/feature_ablation.html



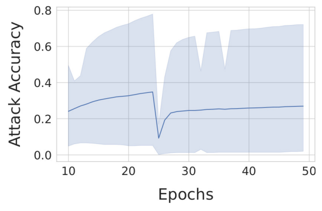
(a) Adult



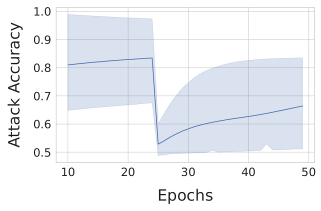
(b) ai4i



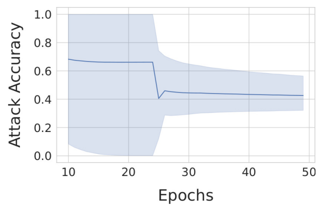
(c) Hepmass



(d) Poqemon



(e) Susy



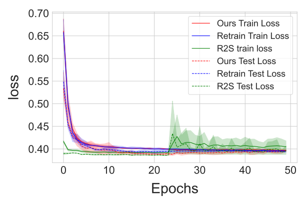
(f) Wine

Figure 8: The MIA attack accuracy (y-axis) of VFU-KD.

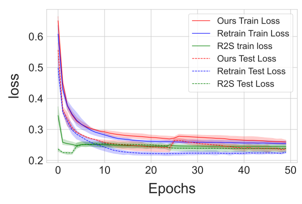
Dataset	RfS		R2S		Ours	
	AUC	F1	AUC	F1	AUC	F1
Adult	0.82	0.81	0.81	0.81	**0.82**	**0.82**
ai4i	0.91	0.93	0.91	0.93	**0.91**	**0.93**
Hepmass	0.86	0.86	0.85	0.85	**0.86**	**0.86**
Poqemon	0.92	0.74	0.92	0.72	**0.93**	**0.77**
Susy	0.80	0.80	0.80	0.80	**0.80**	**0.80**
Wine	0.99	0.99	0.99	0.99	**0.99**	**0.99**

Table 7: The AUC and F1 score comparison of VFU-GA (Ours) with the retrained-from-scratch (RfS) model and the R2S method for unlearning 5 batches.

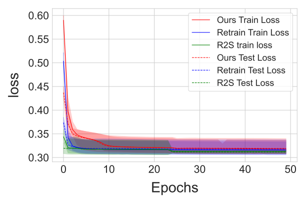
and test loss for unlearning the most important feature and Fig. 7 shows the training and test loss for unlearning the least important feature for all the datasets. Table 5 and Table 6 show the results for AUC score and F1 score. Here as well, in all the datasets, the loss values and utility scores of VFU-KD are benchmark comparable, even better in some cases. Hence, we can say that our approach, VFU-KD, can effectively unlearn passive party as well as feature(s) from passive party. Additional feature unlearning experiments such as unlearning multiple features and most-least important features from each client, are available in the supplementary material.



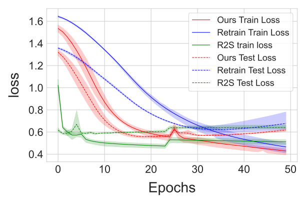
(a) Adult



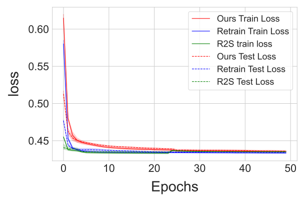
(b) ai4i



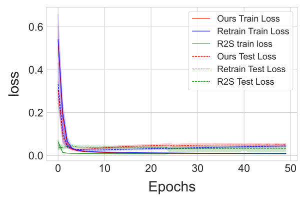
(c) Hepmass



(d) Poqemon



(e) Susy



(f) Wine

Figure 9: The training and test loss of VFU-GA (5 batches) for least important feature, compared to the retrained model from scratch and R2S method.

4.3 Sample unlearning

For sample unlearning, we removed 5 batches from the active model for each dataset. The number of gradient ascent steps was set to 5, chosen arbitrarily. This parameter can be adjusted and increased based on the unlearning requirements of the application. Fig. 9 shows the comparison of training and test loss curves between our approach, retrained from scratch method and R2S method and Table 7 shows the utility scores (F1 and AUC score) when unlearning happened at 25th epoch. The results show that VFU-GA has better utility score than retrained model. Notably, in the case of the Poqemon dataset, VFU-GA demonstrates significantly superior utility. This improvement can be attributed to the robustness introduced by gradient ascent, as models often become more robust following gradient ascent. This characteristic of gradient ascent has been leveraged in previous literature to enhance model robustness, as seen in studies such as [28] and [37]. Additional experimental results for unlearning a batch are available in the supplementary material.

4.4 Auditing VFU-KD

In this section, we evaluate the effectiveness of the unlearning process using a MIA. As discussed in Section 3.1, we train an MIA model with the output logits for 10 epochs both in the presence

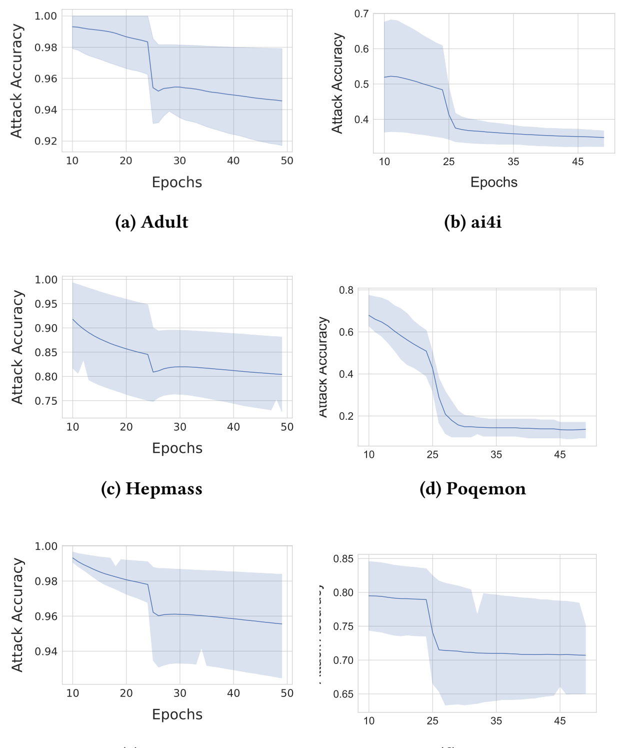


Figure 10: The MIA attack accuracy (y-axis) of VFU-GA.

and absence of *clientA*. The MIA model consists of a single hidden layer comprising 32 neurons. The output layer of the MIA model is a binary classifier which predicts whether *clientA* was present during training or not.

Fig. 8 shows the accuracy of the MIA on the tabular dataset starting from epoch 10 onwards. The results clearly demonstrate a significant drop in MIA accuracy at the 25th epoch, thereby indicating the effect of the unlearning process. Similar results were observed in Fig. 10 on all the datasets for MIA attack accuracy. However, the drop in accuracy can vary with the impact of samples unlearned.

4.5 Limitations

Based on our analysis and the results obtained, we highlight the following limitations of our approach.

(1) *Attack Vulnerability*: Dishonest or honest-but-curious passive parties could potentially exploit the spikes caused by distillation to perform membership inference attacks or gradient based attacks.
(2) *Limited Heterogeneity*: Similar to many existing works in the VFL literature, our approach assumes limited data heterogeneity and that all passive parties are readily available for training, with no stragglers.
(3) *Unlearning Auditing*: For auditing unlearning, we employ a relatively weak membership inference attack (MIA) model. Utilizing a stronger model [14], could yield more insightful results.
(4) *Additional Storage*: Active party stores the communicated embeddings. This might be problematic where active and passive parties are in area which governs different data regulatory laws.

5 Conclusion and Future work

In this paper, we introduced a framework for unlearning in vertical federated learning (VFL), focusing on passive party unlearning and feature unlearning using knowledge distillation, termed VFU-KD, and sample unlearning using gradient ascent, termed VFU-GA. VFL is inherently communication intensive. Thus, an effective unlearning approach should aim to minimize the communication between the active and passive parties. In VFU-KD, the active party is responsible for passive party unlearning, while the respective passive party handles feature unlearning. VFU-KD leverages knowledge distillation for effective model compression and unlearning. On the other hand, since sample unlearning does not require model compression, gradient ascent provides a more computationally efficient option in VFU-GA.

Our approach does not require any communication between active party and passive party for unlearning. However, it requires that the active party stores the communicated embeddings. This is essential in order to not have any communication. We have also proposed a MIA which can be used to audit unlearning in VFL. We have compared VFU-KD, VFU-GA with the gold standard unlearning model i.e., model retrained from scratch and R2S optimization based faster retraining, on 6 tabular datasets, and 2 image dataset. The results demonstrate that, with our approach both active and passive parties can perform unlearning without any significant utility loss.

In our experiments, we employed a simple binary classifier for the membership inference attack (MIA). However, leveraging a more advanced and robust MIA model, such as the one proposed in [4], could potentially yield more insightful and accurate results. Exploring this avenue remains a priority for future work. Additionally, investigating the relationship between the distillation-induced spikes and their susceptibility to membership inference attacks presents an intriguing research direction. Another interesting direction for future work is to develop strategies to reduce the storage overhead for the active party, further enhancing the efficiency and scalability of the approach.

Acknowledgments

This work was carried out during an internship at Ericsson Research and was partially supported by the Wallenberg AI, Autonomous Systems, and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation. The computational resources required for this research were provided by Ericsson and the Berzelius supercomputer provided by National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg foundation.

References

[1] Lamine Amour, Souihi Sami, Said Hoceini, and Abdelhamid Mellouk. 2015. Building a large dataset for model-based QoE prediction in the mobile environment.

In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems.* 313–317.

[2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International conference on artificial intelligence and statistics.* PMLR, 2938–2948.

[3] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. 2021. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP).* IEEE, 141–159.

[4] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP).* IEEE, 1897–1914.

[5] Adam Coates, Andrew Ng, and Honglak Lee. 2011. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics.* JMLR Workshop and Conference Proceedings, 215–223.

[6] Zihao Deng, Zhaoyang Han, Chuan Ma, Ming Ding, Long Yuan, Chunpeng Ge, and Zhe Liu. 2023. Vertical federated unlearning on the logistic regression model. *Electronics* 12, 14 (2023), 3182.

[7] Changyu Dong, Liqun Chen, and Zikai Wen. 2013. When private set intersection meets big data: an efficient and scalable protocol. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security.* 789–800.

[8] Dashan Gao, Sheng Wan, Lixin Fan, Xin Yao, and Qiang Yang. 2024. Complementary Knowledge Distillation for Robust and Privacy-Preserving Model Serving in Vertical Federated Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 19832–19839.

[9] Antonio Ginart, Melody Guan, Gregory Valiant, and James Y Zou. 2019. Making ai forget you: Data deletion in machine learning. *Advances in neural information processing systems* 32 (2019).

[10] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision* 129, 6 (2021), 1789–1819.

[11] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. 2021. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11516–11524.

[12] Anisa Halimi, Swanand Kadhe, Ambrish Rawat, and Nathalie Baracaldo. 2022. Federated unlearning: How to efficiently erase a client in fl? *arXiv preprint arXiv:2207.05521* (2022).

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[14] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)* 54, 11s (2022), 1–37.

[15] Yu Jiang, Jiyuan Shen, Ziyao Liu, Chee Wei Tan, and Kwok-Yan Lam. 2024. Towards efficient and certified recovery from poisoning attacks in federated learning. *arXiv preprint arXiv:2401.08216* (2024).

[16] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. 2023. The UCI machine learning repository. *URL https://archive. ics. uci. edu* (2023).

[17] Alex Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images. (2009). https://api.semanticscholar.org/CorpusID:18268744

[18] Guanghao Li, Li Shen, Yan Sun, Yue Hu, Han Hu, and Dacheng Tao. 2023. Subspace based federated unlearning. *arXiv preprint arXiv:2302.12448* (2023).

[19] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2021. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering* 35, 4 (2021), 3347–3366.

[20] Gaoyang Liu, Xiaoqiang Ma, Yang Yang, Chen Wang, and Jiangchuan Liu. 2021. Federaser: Enabling efficient client-level data removal from federated learning models. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS).* IEEE, 1–10.

[21] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2019. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265* (2019).

[22] Yang Liu, Tao Fan, Tianjian Chen, Qian Xu, and Qiang Yang. 2021. Fate: An industrial grade platform for collaborative learning with data protection. *Journal of Machine Learning Research* 22, 226 (2021), 1–6.

[23] Yang Liu, Yan Kang, Tianyuan Zou, Yanhong Pu, Yuanqin He, Xiaozhou Ye, Ye Ouyang, Ya-Qin Zhang, and Qiang Yang. 2024. Vertical federated learning: Concepts, advances, and challenges. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[24] Linpeng Lu and Ning Ding. 2020. Multi-party private set intersection in vertical federated learning. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom).* IEEE, 707–714.

[25] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics.* PMLR, 1273–1282.

[26] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP).* IEEE, 739–753.

[27] Prayitno, Chi-Ren Shyu, Karisma Trinanda Putra, Hsing-Chung Chen, Yuan-Yu Tsai, KSM Tozammel Hossain, Wei Jiang, and Zon-Yin Shae. 2021. A systematic review of federated learning in the healthcare area: From the perspective of data properties and applications. *Applied Sciences* 11, 23 (2021), 11191.

[28] Othmane Sebbouh, Marco Cuturi, and Gabriel Peyré. 2022. Randomized stochastic gradient descent ascent. In *International Conference on Artificial Intelligence and Statistics.* PMLR, 2941–2969.

[29] Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. 2023. Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[30] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. 2022. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P).* IEEE, 303–319.

[31] Fei Wang, Baochun Li, and Bo Li. 2023. Federated unlearning and its privacy threats. *IEEE Network* (2023), 463–480. https://doi.org/10.1109/MNET.004.2300056

[32] Zichen Wang, Xiangshan Gao, Cong Wang, Peng Cheng, and Jiming Chen. 2024. Efficient Vertical Federated Unlearning via Fast Retraining. *ACM Transactions on Internet Technology* 24, 2 (2024), 1–22.

[33] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. 2021. Machine unlearning of features and labels. *arXiv preprint arXiv:2108.11577* (2021).

[34] Kang Wei, Jun Li, Chuan Ma, Ming Ding, Sha Wei, Fan Wu, Guihai Chen, and Thilina Ranbaduge. 2022. Vertical federated learning: Challenges, methodologies and experiments. *arXiv preprint arXiv:2202.04309* (2022).

[35] Chen Wu, Sencun Zhu, and Prasenjit Mitra. 2022. Federated unlearning with knowledge distillation. *arXiv preprint arXiv:2201.09441* (2022).

[36] Zhaomin Wu, Junyi Hou, Yiqun Diao, and Bingsheng He. 2024. Federated Transformer: Multi-Party Vertical Federated Learning on Practical Fuzzily Linked Data. *arXiv preprint arXiv:2410.17986* (2024).

[37] Dongkeun Yoon, Joel Jang, Sungdong Kim, and Minjoon Seo. 2023. Gradient Ascent Post-training Enhances Language Model Generalization. In *The 61st Annual Meeting Of The Association For Computational Linguistics.*

[38] Chong Yu, Shuaiqi Shen, Shiqiang Wang, Kuan Zhang, and Hai Zhao. 2024. Communication-Efficient Hybrid Federated Learning for E-Health With Horizontal and Vertical Data Partitioning. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[39] Haibo Zhang, Toru Nakamura, Takamasa Isohara, and Kouichi Sakurai. 2023. A review on machine unlearning. *SN Computer Science* 4, 4 (2023), 337.

[40] Fanglan Zheng, Kun Li, Jiang Tian, Xiaojia Xiang, et al. 2020. A vertical federated learning method for interpretable scorecard and its application in credit scoring. *arXiv preprint arXiv:2009.06218* (2020).

[41] Xiangrong Zhu, Guangyao Li, and Wei Hu. 2023. Heterogeneous federated knowledge graph embedding learning and unlearning. In *Proceedings of the ACM web conference 2023.* 2444–2454.

A Hessian utility score

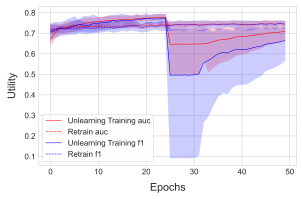
Fig. 11 shows the F1 and AUC scores for client unlearning with $\mathcal{H}^{-1}$. Here as well, it is clear from the results that the utility score eventually converges to the benchmark comparable utility scores in all the cases except Hepmass and Poqemon dataset. We found that, we have better results in the presence of a learning rate than with $\mathcal{H}^{-1}$.

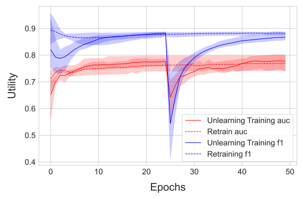
B Additional feature importance results

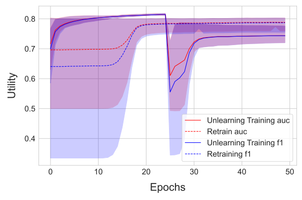
Fig. 14 highlights the importance of each feature using feature ablation. Based on this information, the results for most important and least important feature were given in the main paper.

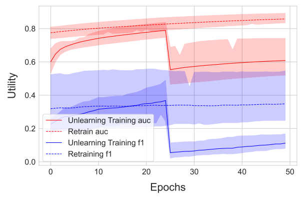
We have also experimented with removing multiple features from the same client, most important feature and least important feature from each clients for wine dataset.

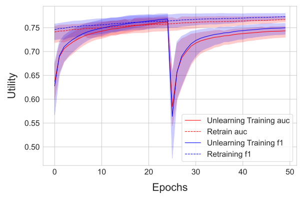
Fig. 12 shows the training and test loss when two most-least important features were unlearning, and most-least important feature from each client were unlearning. Similary, Fig. 13 shows the utility scores for the same. In all the cases, we see that, VFU-KD achieves


(a) Adult


(b) ai4i


(c) Hepmass


(d) Poqemon


(e) Susy

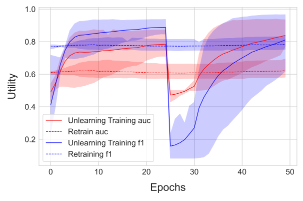

(f) Wine

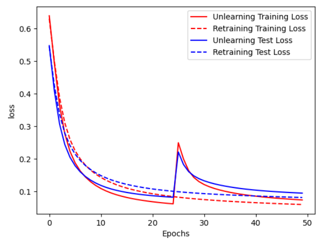
Figure 11: The F1 and AUC scores of VFU-KD with $\mathcal{H}^{-1}$ compared to the retrained model from scratch.

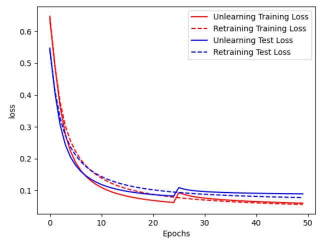
benchmark comparable results, even better than benchmark in some cases.

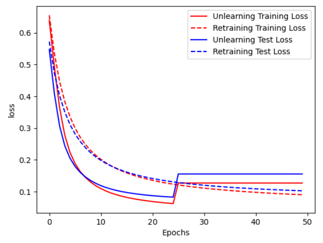
C Additional VFU-GA results

Fig. 9 shows the the comparison of training and test loss between VFU-GA and benchmark model when unlearning 5 batches.

Fig. 15 shows the comparison of training and test loss between VFU-GA and benchmark model. Here as well, the model performs better than the benchmark model after unlearning attributed to the robustness introduced while unlearning. Fig. 16 shows the performance of MIA attack when unlearning 1 batch. We can see the evidence of unlearning with the drop in accuracy.


(a) Two Most Important Features


(b) Two Least Important Features


(c) Most Important Feature from Each Client

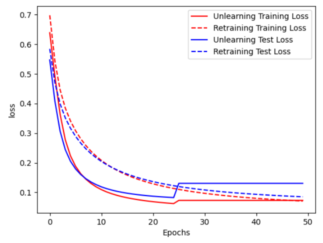
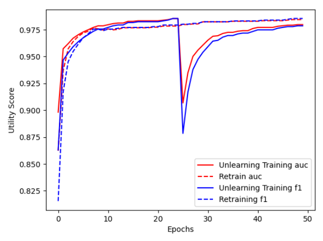
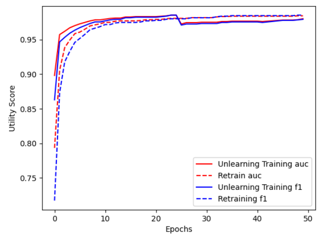
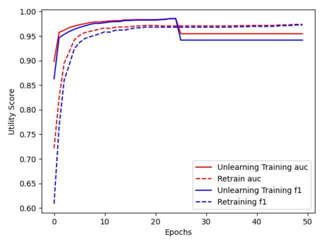

(d) Least Important Feature from Each Client

Figure 12: The training and test loss values of VFU-KD for the wine dataset.


(a) Two Most Important Features


(b) Two Least Important Features


(c) Most Important Feature from Each Client

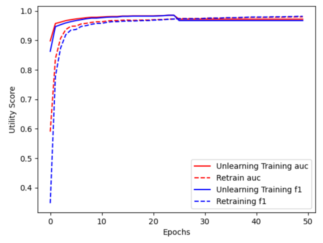
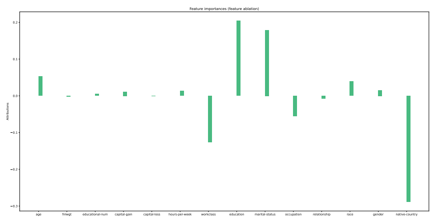
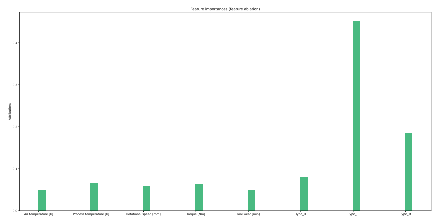

(d) Least Important Feature from Each Client

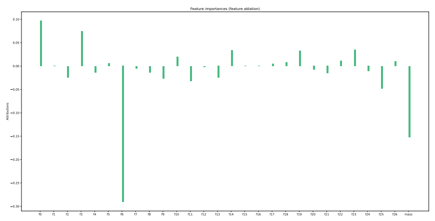
Figure 13: The F1 and AUC score of VFU-KD for the wine dataset.



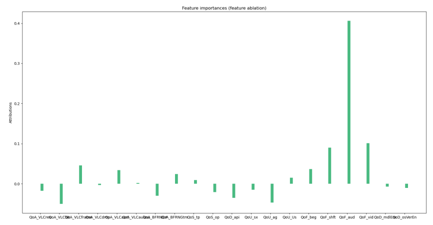
(a) Adult



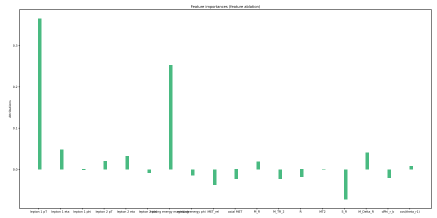
(b) ai4i



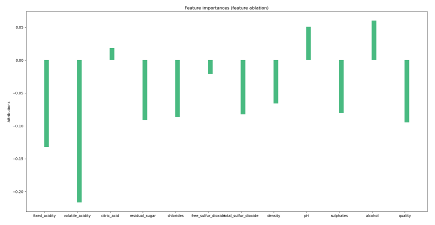
(c) Hepmass



(d) Poqemon

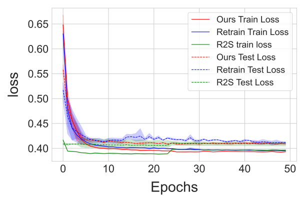


(e) Susy

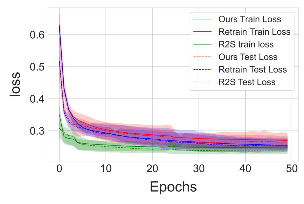


(f) Wine

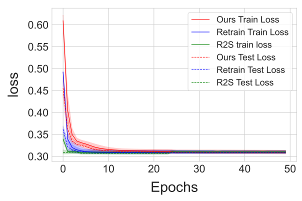
Figure 14: The feature importance plot. Each bar, from left to right, represents the features in order from the first to the last column of the respective dataset.



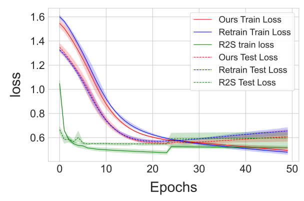
(a) Adult



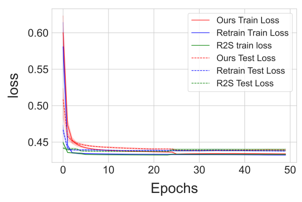
(b) ai4i



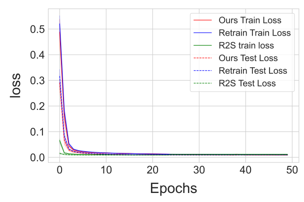
(c) Hepmass



(d) Poqemon



(e) Susy

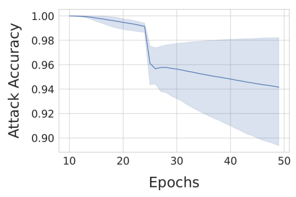


(f) Wine

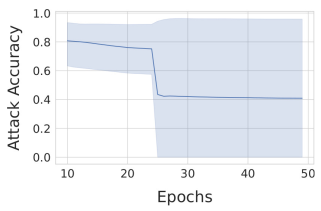
Figure 15: The training and test loss of VFU-GA (1 batch) for least important feature, compared to the retrained model from scratch and R2S method.

Dataset	RfS		R2S		Ours	
	AUC	F1	AUC	F1	AUC	F1
Adult	0.81	0.78	0.80	0.80	**0.81**	0.78
ai4i	0.91	0.93	0.91	0.93	**0.91**	**0.93**
Hepmass	0.86	0.86	0.86	0.86	**0.86**	**0.86**
Poqemon	0.93	0.75	0.93	0.76	**0.93**	**0.76**
Susy	0.80	0.80	0.80	0.80	**0.80**	**0.80**
Wine	0.99	0.99	0.99	0.99	**0.99**	**0.99**

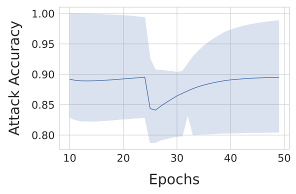
Table 8: The AUC and F1 score comparison of VFU-GA (Ours) with the retrained-from-scratch (RfS) model and the R2S method for unlearning 1 batches.



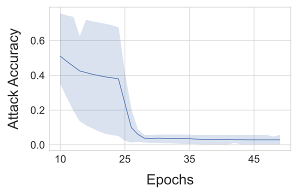
(a) Adult



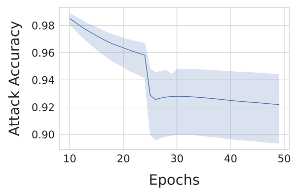
(b) ai4i



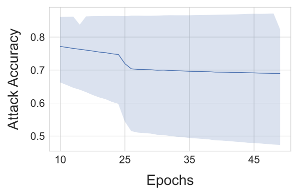
(c) Hepmass



(d) Poqemon



(e) Susy



(f) Wine

Figure 16: The MIA attack accuracy (y-axis) of VFU-GA (1 batch).