Privacy-Preserving Split Learning via Pareto Optimal Search

Xi Yu[1], Liyao Xiang[1][0000−0003−0165−4930]⋆,
Shiming Wang[1], and Chengnian Long[1]

Shanghai Jiao Tong University
yuxi2017, xiangliyao08, my16wsm,longcn@sjtu.edu.cn

Abstract. With the rapid development of deep learning, it has become a trend for clients to perform split learning with an untrusted cloud server. The models are split into the client-end and server-end with features transmitted in between. However, features are typically vulnerable to attribute inference attacks to the input data. Most existing schemes target protecting data privacy at the inference, but not at the training stage. It remains a significant challenge to remove private information from the features while accomplishing the learning task with high utility. We found the fundamental issue is that utility and privacy are mostly conflicting tasks, which are hardly handled by the linear scalarization commonly used in previous works. Thus we resort to the multi-objective optimization(MOO) paradigm, seeking a Pareto optimal solution according to the utility and privacy objectives. The privacy objective is formulated by the mutual information between feature and sensitive attributes and is approximated by Gaussian models. In each training iteration, we select a direction that balances the dual goal of moving toward the Pareto Front and toward the users' preference while keeping the privacy loss under the preset threshold. With a theoretical guarantee, the privacy of sensitive attributes is well preserved throughout training and at convergence. Experimental results on image and tabular datasets reveal our method is superior to the state-of-the-art in terms of utility and privacy.

Keywords: Split learning · Privacy · Pareto optimal.

1 Introduction

Deep learning shows an impressive performance in various fields such as computer vision and natural language processing, benefiting from the complicated model architectures and massive training data. It is often infeasible for data owners to perform the entire end-to-end training with limited computing resources such as mobile devices. A popular solution is the *split learning*, where the model is partitioned into two parts: the client-end and the server-end. Data owners locally encode their private inputs into features with the client-end encoder model and resort to an untrusted server to further process the features and complete

⋆ Corresponding author.

the training or inference tasks. An eavesdropper or the server can infer sensitive attributes of the private inputs from features. The clients would often specify some sensitive attributes, such as gender, age, etc. to remove from the features before sending them to the server. Hence removing sensitive information from the features without affecting the learning tasks is a critical issue.

To resolve the issue, previous approaches have been proposed to either 'hide' the features by crypto techniques such as Homomorphic Encryption(HE) [16] and Secure Multi-Party Computation(SMPC) [24], or by learning-based methods which train the encoder to remove the sensitive attributes from features [6, 8, 9, 13, 25]. The former brings significant computation and communication overhead, especially in deep networks. The latter unfortunately does not guarantee privacy throughout the training, as the sensitive information removal requires the learning w.r.t. the privacy loss to converge.

Our scheme follows the line of learning-based approaches but non-trivially preserves privacy over training. It is inspired by the most recent progress in multi-task learning (MTL), a well-established paradigm for learning multiple correlated tasks. A long-neglected fact in the current adversarial learning literature is that preserving the privacy of sensitive attributes and completing the learning task are mostly conflicting objectives, as the learning task may implicitly require the sensitive attributes and thus it is fairly hard to remove such attributes without hurting the task utility. It is notoriously difficult to balance conflicting goals with the conventional linear scalarization method, mostly used in previous works. Hence we formulate our problem as an MOO, set the client's preference vector over privacy versus utility, and search for an Exact Pareto Optimal (EPO) [10] solution w.r.t. the two objectives to find specific tradeoffs given preferences. Pareto optimal solutions mean that, in MOO, no objective value can be further improved without degrading some others.

The privacy objective is formulated by the mutual information loss, similar to [8, 13]. We transform the intractable mutual information to the Kullback-Leibler Divergence (KL-divergence) between the feature distributions. To calculate the KL-divergence, we assume each feature is sampled from a Gaussian distribution, and adopt a reparametrization trick [1] to make it differentiable. As sampling and computing in the high-dimensional space are costly, we run Principal Component Analysis(PCA) to extract low-dimensional representations for computing the mutual information loss. The utility objective is essentially the learning task loss.

Although a Pareto optimal solution may present a better tradeoff in utility and privacy, it does not guarantee the training stage privacy. We propose to fill the gap by explicitly controlling the optimization trajectory to keep the privacy loss below the tolerance threshold throughout training. Specifically, instead of directly approaching the Pareto Front as in Fig. 1(a), we let the client first pre-train its encoder to reduce the privacy loss below the preset threshold Th, as in Fig 1(b). In each of the following training iterations, we search for an appropriate direction to balance the goals of (1) moving toward the Pareto front and (2) towards the preference vector. The utility gradually improves by a controlled

ascent in the privacy loss until reaching Th. Once the privacy constraint is activated, we continue looking for a direction without increasing the privacy loss; if such a direction cannot be found, the optimization stops at a point closest to the preference vector and Pareto Front. As shown in Fig 1(b), the utility loss steadily decreases in traversing the Pareto Front while the trajectory is consistently below the privacy threshold, protecting sensitive attributes throughout the training.

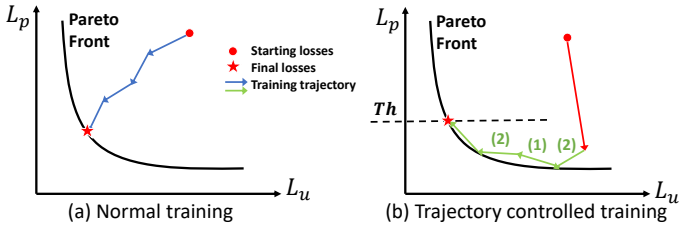

Fig. 1. Trajectory-Controlled EPO compared to normal training. Our method guarantees privacy during training by tracing the Pareto front and curbing the privacy loss (L_p) below Th. L_u refers to the utility loss. The indices (1) and (2) in (b) refer to different optimization steps.

Highlights of our contributions are as follows. First, we innovatively use MOO to handle conflicting tasks in privacy-preserving split learning problem and seek an EPO solution for a better tradeoff between utility and privacy. Second, we propose a trajectory-controlled optimization approach by tracing the utility-privacy Pareto Front with the privacy constraint, thereby protecting sensitive attributes during training with theoretical guarantees. Finally, experiments on image and tabular data verify our approach, showing superiority in utility and privacy against training and inference attacks, compared to the state-of-the-art.

2 Related Works

We focus on prior works defending against attribute inference attacks in the split learning scenario.

2.1 Adversarial Training-based Defense

One line of works are adversarial training-based defense [9, 22, 23], which simulates a game between the attacker and defender with conflicting privacy goals. Specifically, the client-end's encoder pits against a simulated attacker by misleading it into making the wrong predictions about the sensitive attributes. The encoder is expected to generate features that preserve the client's attribute privacy as the adversarial training ends. However, Pittaluga et al. [15] point out

that adversarial training can hardly reach a theoretical stationary point, so such defense is not robust against adaptive attackers that continue to update its classifier after the defense strategy is fixed. Label Flip Loss [15] proposes to mitigate the problem by reducing the attacker's prediction confidence, despite the prediction being correct or not. However, the empirical improvement is minor overall.

2.2 Information Theoretical Defense

An alternative approach is to limit the sensitive information contained in the client's features, eliminating the privacy threat from the root [8,12,13,25]. Specifically, they try to reduce the mutual information between features and sensitive attributes to preserve user privacy, while maximizing the mutual information between features and utility label. However, the aforementioned works share two common defects. First, they use linear scalarization to balance privacy and utility goals, which performs poorly when handling conflicting goals. Additionally, these methods only ensure feature privacy after the training convergences but leave training privacy an open question. In comparison, our method searches for the Pareto optimal solution instead to achieve a specific tradeoff and protects the client's data privacy during training by trajectory-controlled optimization.

3 Preliminaries

3.1 Split Learning

Split learning [3] enables multiple parties to train a model collaboratively without sharing raw data. As shown in Fig. 2(a), the whole deep learning model is partitioned into an encoder f_θ assigned to the client-end and a subsequent classifier g_Φ deployed on the server-end. During one training epoch, the clients forward their private input X through the encoder and transmit the feature $f_\theta(X)$ to the server. The server continues to forward the feature through its classifier and runs back-propagation to compute gradient updates for g_Φ and then f_θ.

Split learning greatly reduces the client's privacy risk and computation cost. The clients transmit the smashed feature instead, which prevents raw input exposure and saves transmission bandwidth. In addition, the server shares the training burden, which allows clients with limited computing resources such as Internet of Things(IoT) devices to participate in deep learning.

3.2 Multi-Task Learning and Multi-Objective Optimization

MTL allows several tasks to be learned at the same time in a single model. A popular training approach is to minimize a weighted sum of losses for different tasks, referred to as *linear scalarization*. Despite the concise form, this method often fails to handle conflicting tasks common in real-world applications [10].

To better balance conflicting tasks, Sener et al. [18] propose to solve the MTL problem by MOO. Instead of pursuing optimality for all tasks simultaneously,

MOO attains optimal tradeoffs called *Pareto optimal solutions*. Each solution is a non-dominated tradeoff point where no loss can be further reduced without deteriorating other objectives. Together these solutions compose the *Pareto Front* of the learning problem. The state-of-the-art approach of MOO with high preciseness and efficiency is the Exact Pareto Optimal (EPO) search [10]. Given a vector $\boldsymbol{r} = (r_1, \cdots, r_n)$ indicating the client's preference for different objectives, EPO search finds the solution which is in the intersection of the ray $\boldsymbol{r}^{-1} = (1/r_1, \cdots, 1/r_n)$ and the Pareto Front if it exists.

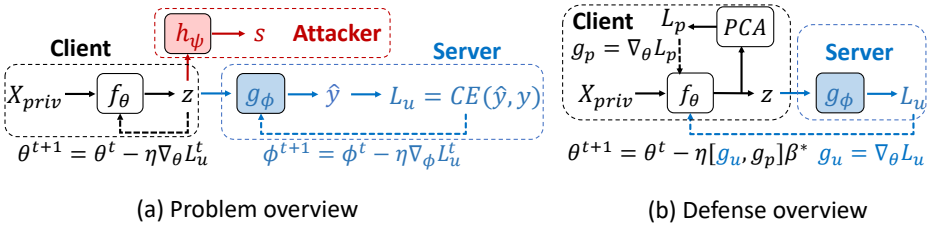

Fig. 2. Problem formulation and defense overview. $\boldsymbol{g}_u, \boldsymbol{g}_p$ denote the gradient of utility loss, and privacy loss, respectively and β^* is the solution to (12). The client, server, and attacker are represented by black, blue, and red respectively. The server can also be an attacker.

4 Problem Formulation and Threat Model

We focus on the typical split learning scenario where multiple clients train a deep learning model collaboratively. Due to privacy concerns, any raw data or feature transmission that potentially exposes the clients' sensitive information should be avoided during the model *training* and *inference* stages. We formally show the problem setting in Fig 2(a). The three parties involved are the *client*, *server*, and *attacker*. A deep learning model is split into a client-end encoder f_θ and a server-end classifier g_Φ. The clients encode their private input X_{priv} into feature $z = f_\theta(X_{\text{priv}})$ before sending them to the server for model update (training) or label prediction (inference).

Unfortunately, transmitting features is still prone to privacy leakage. An honest-but-curious inside attacker in the server or a man-in-the-middle attacker receives the feature z and can infer clients' sensitive attributes s. This is known as the *attribute inference attack*. Previous work shows that the attacker is widely successful in inferring sensitive attributes including gender, age, race, etc. [7, 17, 20]. The attribute inference attack works because a model trained for a simple task tends to overlearn attributes that are irrelevant to the objective task [19].

Next, we instantiate representative attack methods categorized into training and inference stages, according to whether the attacker utilizes the training

trajectory of the model. In both cases, we assume the attacker owns an auxiliary dataset X_{aux} following a similar distribution to the clients' X_{priv} and marked with sensitive attribute labels. The attacker trains a decoder model h_ψ over X_{aux} and known sensitive attribute labels.

Training stage attacks including Decoding Simulator Attack (DSA) [21] and Data Poison Attack (DPA). The DSA attacker intercepts the clients' private feature $f_\theta(X_{\text{priv}})$ at each training epoch. It trains a shadow encoder $\tilde{f}$ with X_{aux} so that the shadow features $\tilde{f}(X_{\text{aux}})$ and real features are hard to distinguish by a discriminator model. That is, the shadow encoder approximates the feature distribution of the real encoder from its training trajectory. Simultaneously, the attacker trains the decoder model h_ψ to infer sensitive attributes with $(\tilde{f}(X_{\text{aux}}), s)$. Besides attaining the encoder's training trajectory, DPA attackers can further insert poison samples X_p into clients' training set. The attacker intercepts $f_\theta(X_p)$ at each training epoch as the clients upload their features. It then trains h_ψ with $(f_\theta(X_p), s)$.

Inference stage attacker is able to query the trained model f_{θ^*} multiple times to collect the corresponding feature of X_{aux} and trains the attack model with $(f_{\theta^*}(X_{\text{aux}}), s)$.

For ease of understanding, we list all notations in Appendix A.

5 Methodology

In face of attribute inference attacks, we propose training the encoder with the following goals: 1) *Privacy*: attackers fail to infer the sensitive attribute s from features during training or inference. 2) *Utility*: the features should complete the original learning tasks with high performance.

Intuitively, the feature $z = f_\theta(X_{\text{priv}})$ should contain little information about s while much information about the utility label y. By describing the utility and privacy objectives as mutual information losses $I(\cdot;\cdot)$, we formulate the problem as one to maximize the utility while curbing the privacy leakage. Expressed by the conventional linear scalarization, the objective is

$$\max_{\theta} \quad I(z;y) - \lambda I(z;s) \tag{1}$$

where λ controls the tradeoff between privacy and utility. We follow previous work [4, 13] to transform objective (1) into differentiable utility loss $\mathcal{L}_u$ and privacy loss $\mathcal{L}_p$ in §5.1. The difficulty lies in balancing the conflicting goals of privacy and utility, which have been neglected in most previous works. We argue that with careful manipulation of the optimization trajectory, the privacy leakage through z could be restrained, throughout the training process and at the convergence. The trick is to formulate the problem as a MOO and perform a controlled EPO search and the detail is provided in §5.2, i.e., each update step of the client's encoder is selected in a direction where the privacy loss is restricted below a given level while the utility loss steadily decreases. Finally, we present the overall privacy-preserving split learning algorithm with guaranteed privacy in §5.3. Our defense overview is shown in Fig 2(b).

5.1 Privacy and Utility Losses

Since it is infeasible to formulate the mutual information of high-dimensional variables in (1), as a proxy, we resort to maximizing the tractable lower bound $\mathcal{L}$ of $I(z;y)$ and minimizing the upper bound $\mathcal{U}$ of $I(z,s)$ as in [13]. The lower and upper bounds are as follows and the detailed derivation can be found in Appendix B.

$$\mathcal{L} = H(y) + \max_{\phi} \mathbb{E}_{z,y} \log q_\phi(y|z), \quad \mathcal{U} = \sum_a \sum_{b \neq a} p(s_a)p(s_b)D_{KL}[p(z|s_a)||p(z|s_b)]. \tag{2}$$

In $\mathcal{L}$, $H(y)$ denotes the entropy of y, and $q(y|z)$ is a conditional distribution. For $\mathcal{U}$, p denotes the probability distribution of sensitive attributes (with a, b denoting the different actual values of the attribute), and D_{KL} means the KL-divergence. Since $H(y)$ is a constant, we only need to optimize the second term of $\mathcal{L}$, which is equivalent to minimizing the cross-entropy utility loss $L_u = CE(g_\phi(f_\theta(x)), y)$.

The upper bound $\mathcal{U}$ is still too complicated to compute and requires the approximation of output distributions. Considering the deterministic feature of a given input, we add a zero-mean Gaussian noise to the feature and approximate $p(z|s_k)$ with a Gaussian Mixture Model (GMM): $p(z|s_k) = \frac{1}{N_k} \sum_{i=1}^{N_k} p(z|x_i)$, where k represents the specific sensitive attribute value and N_k denotes the corresponding number of features. The KL-divergence in (2) hence becomes KL-divergence between two GMMs. However, it is still too complicated and thus we further approximate the GMM with Gaussian distribution $\hat{p}$ as in [4] and the upper bound $\mathcal{U}$ becomes a differentiable privacy loss

$$L_p = \frac{1}{N^2} \sum_a \sum_{b \neq a} N_a N_b D_{KL}[\hat{p}(z|s_a)||\hat{p}(z|s_b)] \tag{3}$$

where N is the number of samples. Hence following (1), our objective now becomes minimizing the utility loss L_u and the privacy loss L_p at the same time.

Nevertheless, if the encoded features are high-dimensional for ML models, it will bring heavy computation overhead when calculating GMM losses. In this case, we adopt PCA to reduce the dimension of $f_\theta(x)$ to $f'_\theta(x)$. Due to the dimension reduction, $f'_\theta(x)$ no longer follows GMM but we simply estimate them by Kernel Density Estimation (KDE) with Gaussian kernels [2]. The features z transmitted to the server are still computed using $f_\theta(x)$.

5.2 Trajectory-Controlled Pareto Optimal Search

As aforementioned, linear scalarization cannot handle conflicting objectives well. Protecting data privacy and preserving data utility simultaneously is just such a conflicting problem. Additionally, existing MOO-based methods can only ensure feature privacy in the inference stage and training privacy is out of scope. In this section, we will demonstrate in detail how we take advantage of trajectory-controlled Pareto optimal search for guaranteeing privacy throughout training.

We begin by analyzing the relationship between privacy loss and the attack success rate. As mentioned in [25], the theoretical connection between attribute inference accuracy and mutual information is as follows:

$$\Pr[\hat{s} = s] \leq 1 - \frac{H(s)}{\log |\mathcal{S}|} + \frac{I(z; s) + \log 2}{\log |\mathcal{S}|}, \tag{4}$$

where $\mathcal{S}$ means the space of all possible values of s. Hence with a smaller $I(z; s)$, the upper bound of the attack success rate is lower. It means that if L_p is kept at a low level throughout the training process, the privacy of the sensitive attribute is preserved.

To establish control over L_p, we resort to a gradient-based MOO algorithm – Trajectory-Controlled Pareto Optimal(TCPO) search, which traces the Pareto Front by gradually decreasing utility loss with a controlled ascent of the privacy loss. Here we consider a more general case where the number of losses are more than two. Let $\boldsymbol{l} = (l_1, \cdots, l_m) \succeq \mathbf{0}$ represents the non-negative losses of m different utility and privacy tasks. We define a preference order by the preference vector $\boldsymbol{r} \in \mathbb{R}_+^m$ that, for any $i, j \in [m]$, if $r_i \geq r_j$, then $l_i \leq l_j$, suggesting task i is more important than task j. An EPO solution [10] is defined w.r.t. $\boldsymbol{r}$ as:

$$\mathcal{P}_r = \{\theta^* \in \mathcal{P} | r_1 l_1^* = r_2 l_2^* = \cdots = r_m l_m^*\}, \tag{5}$$

where $l_i^* = l_i(\theta^*)$ at convergence, and $\mathcal{P}$ denotes the set of Pareto optimal solutions. The definition suggests that the solution is the intersection between the Pareto Front and the ray $\boldsymbol{r}^{-1} := (1/r_1, 1/r_2, \cdots, 1/r_m)$.

Since our goal is to suppress the privacy leakage, we set the preference vector as follows: $r_i = 1$ if l_i is the utility loss, otherwise $r_i = 0$. By definition, the ideal solution satisfying (5) can never be obtained, and thus the utility loss would always be decreasing by the criteria. Given $\boldsymbol{r}$, the optimization process includes: initializing the encoder so that the privacy constraint is satisfied, and two iterative steps of approaching the $\boldsymbol{r}^{-1}$ ray and the Pareto Front. Suppose the gradient matrix of the losses is $G = [\boldsymbol{g}_1, \boldsymbol{g}_2, \cdots, \boldsymbol{g}_m]$, and $C = G^T G$. In each step, we select the update direction of θ as $\boldsymbol{d} = G\boldsymbol{\beta}$, where $\boldsymbol{\beta}$ denotes the linear weights of gradients. In the following, we detail each of the steps.

At the initialization step, we locally pre-train the encoder to minimize the privacy loss under the threshold Th before uploading any feature. As soon as the client begins to upload, the features are vulnerable to attacks. Hence it is important to restrain the privacy leakage from the first round of updates.

In the following optimization step, we apply a non-uniformity loss to approach the $\boldsymbol{r}^{-1}$ ray, which is expressed by the deviation from the $\boldsymbol{r}^{-1}$ ray [11]:

$$\mu_r(\boldsymbol{l}) = \mu(\boldsymbol{l}, \boldsymbol{r^{-1}}) = 1 - \frac{\langle \boldsymbol{l}, \boldsymbol{r^{-1}} \rangle^2}{||\boldsymbol{l}||^2 ||\boldsymbol{r^{-1}}||^2}. \tag{6}$$

It is easy to verify that μ is always positive by Cauchy-Schwarz inequality. Letting $c = \frac{2\langle \boldsymbol{l}, \boldsymbol{r^{-1}} \rangle}{||\boldsymbol{l}||^3 ||\boldsymbol{r^{-1}}||} (c > 0)$, the gradient direction of μ with respect to $\boldsymbol{l}$ is:

$$\frac{\partial \mu_r(\boldsymbol{l})}{\partial \boldsymbol{l}} = c\left(\frac{\langle \boldsymbol{l}, \boldsymbol{r^{-1}} \rangle}{||\boldsymbol{l}|| ||\boldsymbol{r^{-1}}||} \boldsymbol{l} - \frac{||\boldsymbol{l}||}{||\boldsymbol{r^{-1}}||} \boldsymbol{r^{-1}}\right) \triangleq c\boldsymbol{a}, \tag{7}$$

where c is a constant, and $\boldsymbol{a}$ is the anchor direction. The derivative of μ with respect to θ is

$$\frac{\partial \mu_r(\boldsymbol{l})}{\partial \boldsymbol{\theta}} = \frac{\partial \mu_r(\boldsymbol{l})}{\partial \boldsymbol{l}} \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{\theta}} = c\boldsymbol{a}^T G^T. \tag{8}$$

With simple derivation, we have the following claim: the loss vector $\boldsymbol{l}$ and the anchor direction $\boldsymbol{a}$ are always orthogonal, i.e., $\boldsymbol{a}^T\boldsymbol{l} = 0$ and $\boldsymbol{a}$ is scale-invariant to $\boldsymbol{r}^{-1}$. Also, in minimizing μ, the following lemmas hold:

Lemma 1. *([11]) If the direction $\boldsymbol{d}$ satisfies $\boldsymbol{a}^T G^T \boldsymbol{d} \geq 0$, then there exists a step size $\eta_0 > 0$ such that:*

$$\mu_r(\boldsymbol{l}(\boldsymbol{\theta} - \eta \boldsymbol{d})) \leq \mu_r(\boldsymbol{l}(\boldsymbol{\theta})), \quad \forall \eta \in [0, \eta_0]. \tag{9}$$

We can find such a $\boldsymbol{d}$ in the descent direction of μ_r by solving the following quadratic programming problem:

$$\min_{\boldsymbol{\beta} \in [-1,1]^m} ||C\boldsymbol{\beta} - \boldsymbol{a}||_2^2. \tag{10}$$

The range of $\boldsymbol{\beta}$ is set to reduce the search space. Formally, we have

Lemma 2. *The direction $\boldsymbol{d}$ found by* (10) *satisfies $\boldsymbol{a}^T G^T \boldsymbol{d} \geq 0$.*

The proof of the lemma can be found in Appendix C. The lemma suggests that solving (10) would always yield a direction which does not increase μ_r in the worst case ($\boldsymbol{a}^T G^T$ is in the same hyperplane with $\boldsymbol{d}$). Combined with Lemma 1, we could always find a direction to approach the preference ray $\boldsymbol{r}^{-1}$ by solving (10). However, searching for a direction to decrease the non-uniformity loss is not enough. We need to keep the privacy loss under a threshold, meaning that if some privacy loss L_p ever violates the privacy constraint, the loss should be prevented from further increasing. In that case, the gradient of L_p — $\boldsymbol{g}_p$ should be in the same hyperplane of $\boldsymbol{d}$, i.e., $\boldsymbol{g}_p^T \boldsymbol{d} \geq 0$.

Apart from approaching the preference ray, it is required to find a direction $\boldsymbol{d}$ to decrease losses of all tasks, meaning that $\boldsymbol{d}$ preferably agrees with the descent directions of all losses: $G^T\boldsymbol{d} \succeq \boldsymbol{0}$. To search such a $\boldsymbol{d}$ is to solve

$$\min_{\boldsymbol{\beta} \in [-1,1]^m} ||C\boldsymbol{\beta} - \boldsymbol{l}||_2^2, \quad \text{s.t.} \quad \frac{C\boldsymbol{\beta}}{||C\boldsymbol{\beta}||} = \frac{\boldsymbol{l}}{||\boldsymbol{l}||}. \tag{11}$$

Formally, we have

Lemma 3. *If the gradient matrix G is full rank, then the direction $\boldsymbol{d}$ solved by* (11) *satisfies $G^T\boldsymbol{d} \succeq \boldsymbol{0}$ and $\boldsymbol{a}^T G^T \boldsymbol{d} = 0$.*

The proof is in Appendix D. Hence by solving (11), we find a direction to decrease all losses without increasing the non-uniformity loss. The former is indicated by $G^T\boldsymbol{d} \succeq \boldsymbol{0}$ and the latter is derived from Lemma 1. In practice, G is a full rank matrix in most cases because few targeted tasks are highly related to each other.

To sum up, we initialize the local encoder so that the privacy leakage falls under a given threshold. Then we iteratively perform two steps of descent. If

the current losses are near the Pareto front, we decrease the non-uniformity loss to approach the preference ray. Otherwise, we decrease the task losses to draw close to the Pareto front. Meanwhile, we use an indicator function $\mathbb{1}_{Th}$ to denote the privacy loss surpassing a given threshold Th and $\mathbb{1}_{opt}$ to indicate a Pareto optimal point is reached. Our optimization is to seek the linear weights

$$\begin{aligned} \boldsymbol{\beta}^* = \arg \min_{\boldsymbol{\beta} \in [-1,1]^m} & ||C\boldsymbol{\beta} - \boldsymbol{a}\mathbb{1}_{opt} - \boldsymbol{l}(1 - \mathbb{1}_{opt})||_2^2, \\ \text{s.t.} \quad & (1 - \mathbb{1}_{opt})(\frac{C\boldsymbol{\beta}}{||C\boldsymbol{\beta}||} - \frac{\boldsymbol{l}}{||\boldsymbol{l}||}) = \mathbf{0}, \\ & \boldsymbol{g}_p^T G\boldsymbol{\beta}\mathbb{1}_{Th} \geq 0. \end{aligned} \tag{12}$$

By solving (12), we obtain the update direction for the local encoder. Different from direct gradient descent, the privacy loss is in controlled ascent from time to time to minimize the non-uniformity loss, but is kept under the threshold by the inequality constraint of (12). Different from the conventional Pareto optimal search which merely ensures the optimality at convergence, our method keeps track of the Pareto front and thus obtains a much lower privacy loss throughout the optimization, as illustrated in Fig. 1.

Theorem 1. *Letting $\theta \in \mathcal{P}$ be a Pareto optimal solution in the optimization, the direction $\boldsymbol{d} = G\boldsymbol{\beta}^*$ found by* (12) *is not* $\mathbf{0}$ *if the privacy constraint is not activated. Meanwhile, the non-uniformity loss μ keeps decreasing.*

The proof of Thm. 1 is found in Appendix E. According to Thm. 1, our optimization would not stop prematurely at a local Pareto optimal solution. Instead, it keeps searching for new Pareto optimal solutions with a smaller non-uniformity loss without violating the privacy constraint. If the training process activates the privacy constraint, denoted by $\mathcal{F} = \{C\boldsymbol{\beta} | \boldsymbol{g}_p^T G\boldsymbol{\beta} \geq 0\}$, and if the orthogonal projection of $\boldsymbol{a}$ on $\mathcal{S} \cap \mathcal{F}$ is not $\mathbf{0}$, then (12) can still find a direction $\boldsymbol{d}$ to decrease μ, otherwise the optimization terminates.

5.3 Privacy-Preserving Training

Our proposed algorithm is detailed in Alg. 1. Before uploading any feature to the cloud, the client locally updates its encoder to minimize the privacy loss until meeting the privacy constraint (Line 2-7). The starting point obviously satisfies the privacy requirement. At each iteration of training (Line 9-15), the client samples features and send them to the cloud. The cloud feeds feature z into its model and sends back the error gradient w.r.t. the utility loss to the client. Backpropagating both utility loss and privacy loss through the encoder, the client obtains $\boldsymbol{g}_u$ and $\boldsymbol{g}_p$. Given the current gradient $G = (\boldsymbol{g}_u, \boldsymbol{g}_p)$, the client searches the descent direction $\boldsymbol{d} = G\boldsymbol{\beta}$ by (12). Finally, the client's encoder gets updated. The key to preserving privacy for the training data lies in Line 15 which searches the descent direction. The direction is chosen to reach the preferred utility and privacy at convergence while keeping the leakage under the tolerance threshold.

Computation complexity. The most time-consuming part of our algorithm lies in the computation of the privacy loss L_p which is dominated by the feature dimension D and the number of private attributes M, totaling $O(D^3M^2)$.

Algorithm 1: Privacy-Preserving Learning with Trajectory Control

Input: Training dataset $D_{train} = \{x_i, y_i, s_i\}_{i=1}^N$, local encoder f_θ, cloud classifier g_ϕ, noise standard deviation σ, number of training iterations T, privacy loss threshold Th

Output: encoder f_θ, cloud classifier g_ϕ

Initialize θ, ϕ
while $L_p > Th$ **do**
  Sample a minibatch b from D_{train}
  Send each $x \in b$ to the encoder f_θ to get $f_\theta(x)$
  Compute the privacy loss L_p by (3) given $\mathcal{N}(f_\theta(x), \sigma^2)$
  Update θ to minimize L_p
end
for $t \in [1, T]$ **do**
  Sample a minibatch b from D_{train}
  Send each $x \in b$ to encoder f_θ to get $f_\theta(x)$
  Run PCA for $f_\theta(x)$ to obtain $f'_\theta(x)$ and use KDE for distributions estimation
  Compute privacy loss L_p by (3) using estimated distributions
  Sample $z \sim \mathcal{N}(f_\theta(x), \sigma^2)$ and send z to cloud classifier g_ϕ
  Cloud: update ϕ w.r.t. L_u and send back the error gradients
  Backpropagate L_u and L_p through the encoder to acquire the gradient matrix G
  Calculate linear coeffients β by (12) and update θ using Adam optimizer
end

6 Experiments and Evaluation

Through experiments on a variety of datasets and tasks, we aim to answer the following questions:

Q1: Is our split learning defense effective against attribute inference attacks? How does it protect sensitive attributes at the training and inference stages?
Q2: Does our defense maintain utility for training and inference tasks?
Q3: Does our defense improve the privacy-utility tradeoff compared to existing methods?

6.1 Experimental Setup

Our algorithm is implemented on PyTorch 1.12.0 and all experiments are done on Intel(R) Xeon(R) Gold 6240C with GPU GeForce RTX 3090.

Datasets and Tasks. We choose representative datasets and tasks to evaluate our approach. In the category of image data, we use the widely-adopted CelebA containing over 200K images labeled with 40 binary attributes to perform attribute classification. We also select two tabular datasets, the Health Heritage dataset containing 55K medical records, and the Census Income (Adult) dataset containing 48K personal information records with 14 attributes, for classification tasks. Each dataset is split into three non-overlapping parts: the client's training dataset X_{train}, the client's testing dataset X_{test}, and the attacker's auxiliary dataset X_{aux} on a 4:1:2 ratio.

The privacy and utility attributes of different datasets are listed in Table 1. The 'Age' attribute has nine possible values, the 'Relationship' attribute has six possible values while all others are binary attributes.

Table 1. Privacy and utility attributes for different datasets.

Dataset	CelebA	Health	Adult
Privacy attribute	Male/Attractive/WearingLipstick	Age/Gender	Sex/Relationship
Utility attribute	Similing	Charlson	Income>50K

Attack Methods. We implement three different attribute inference attack methods on X_{aux}. Two training stage attacks include DSA [21] and DPA as described in Sec. 4. In comparison, DSA is relatively weaker and it merely succeeds on the image data but fails on the tabular data. Hence we only launch DSA to the image data in our experiments. DPA implementation follows the convention by poisoning 5% of the training data. The inference phase attack is implemented as a black-box attack as detailed in Sec. 4, referred to as 'Basic' later.

Table 2. Tradeoff parameters for the baselines.

Dataset	Private Attribute	AdvTrain (ω)	AdvTrain+Label Flip (ω)	Infocensor (λ)
CelebA	Male	$\{1, 3, 5\}$	$\{0.5, 1, 3\}$	$\{0.1, 0.3, 0.4\}$
	Attractive	$\{1, 3, 5\}$	$\{0.5, 1, 3\}$	$\{0.1,\ 0.3,\ 0.4\}$
	WearingLipstick	$\{1, 3, 5\}$	$\{0.5, 1, 3\}$	$\{0.1,\ 0.3,\ 0.4\}$
Health	Age	$\{0.5, 1, 3, 5\}$	$\{0.1, 0.5, 1\}$	$\{0.01, 0.05, 0.1\}$
	Gender	$\{1, 3, 10, 20\}$	$\{3, 5, 10\}$	$\{0.05,\ 0.1,\ 0.3\}$
Adult	Sex	$\{1, 3, 10\}$	$\{1, 3, 5\}$	$\{0.1, 0.3, 0.5, 0.8\}$
	Relationship	$\{1, 3, 5\}$	$\{1, 2, 3\}$	$\{0.1, 0.3, 0.5, 0.8\}$

Baselines. We compare our approach with three state-of-the-art privacy-preserving methods against black-box attribute inference attacks: 1) Adversarial training [9] (ADV) pits against a simulated adversary who tries to infer the sensitive attribute information from the intermediate features until the simulated adversary fails to do so. 2) Adversarial training with label flip [15] (ADV2) improves upon the conventional adversarial training by treating the opposite of the label that the current attacker predicts as the true label. The encoder seeks

to reduce the attacker's confidence in classifying sensitive attributes and thus weakening its capability. Both 1) and 2) control the utility-privacy tradeoff with a weight hyperparameter ω. 3) Infocensor [25] is an information theoretical approach that minimizes the mutual information between the features and sensitive attributes. It controls the utility-privacy tradeoff with hyperparameter λ. Since all three baselines present different tradeoffs at varied hyperparameters, for a comprehensive view, we test the methods under a wide range of ωs and λs as listed in Table 2.

Evaluation Metrics. The utility metric is the classification accuracy whereas the privacy metric is the attack accuracy of private attributes. For training data privacy, the highest attack accuracy throughout the entire training process is recorded. For testing data privacy, we let the attacker infer the private attribute from the intermediate feature and report the mean attack accuracy on the test set.

Models and Hyperparameters. For tabular datasets, we employ a two-layer Multi-Layer Perceptron (MLP) as the encoder, and a three-layer MLP for the cloud classifier. For the image dataset, we use a convolutional neural network that consists of a three-convolutional-layer encoder and a cloud classifier with one convolutional layer followed by four fully-connected layers. We choose the attack decoder's network structure the same as the cloud classifier for the three kinds of attacks. Additionally for DSA, the shadow encoder shares the same structure with the encoder while the discriminator contains two convolutional layers and two residual blocks following [14]. We train all the models using the Adam optimizer with the learning rate set to 0.001 by default. The framework-specific hyperparameters for the three baselines and our method are tuned individually for optimal performance. For our method, privacy thresholds Th are set to 0.005, 0.003, and 0.001, and σ is set to 10 on CelebA and 1 on tabular datasets.

6.2 Effect of Trajectory-Controlled Pareto Optimal Search

We verify the effectiveness of our design against the conventional linear scalarization objective in training CelebA. The same utility and privacy losses are adopted in both defense objectives. For linear scalarization, the training objective is $\min L_u + \lambda L_p$. Results of the best privacy performance and within $1-2\%$ utility decline are reported in Tab. 3. As it shows, our method not only achieves better privacy (lower attack success rate) against basic attacks in the inference stage, but also is superior to linear scalarization against DSA and DPA in the training stage. For a closer inspection, we depict the training loss curves for the two methods in Fig. 3. It is clear that with a similar final utility level, our method maintains a low privacy loss throughout the training. It can also be interestingly observed that, the final privacy level of ours always stays close to Th. It is a verification to the contrapositive of Thm. 1 that, if $\boldsymbol{d} = \mathbf{0}$, the privacy constraint is activated, i.e., at convergence, the privacy level is always around Th.

Table 3. Utility and privacy results (%) on CelebA. 'Linear' refers to the linear scalarization objective being used in privacy-preserving split learning.

CelebA	Male				Attractive				WearingLipstick			
	Utility	Basic	DSA	DPA	Utility	Basic	DSA	DPA	Utility	Basic	DSA	DPA
No defense	92.27	93.41	84.42	90.42	92.27	79.12	74.83	77.17	92.27	89.91	83.65	88.18
Linear	91.57	62.54	59.19	61.39	91.28	62.99	54.42	61.76	91.45	64.02	53.21	62.93
Ours	91.29	61.42	56.44	58.17	91.22	61.53	53.26	58.18	91.41	63.14	52.68	61.09

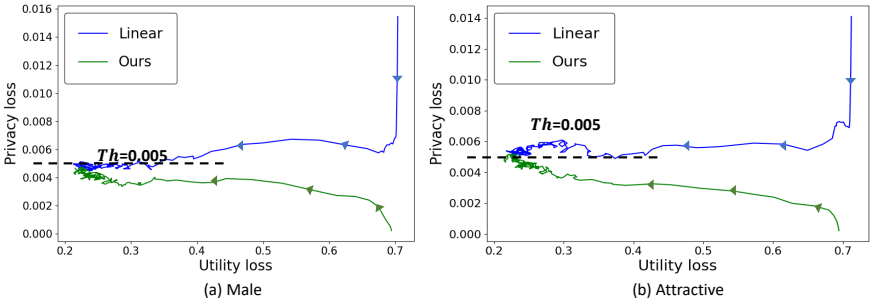

Fig. 3. Training losses trajectory on CelebA. Arrows indicate the iteration sequence. Linear scalarization weight $\lambda = 0.5$. We use uniform filter1d with window size 3 to smooth all the curves.

6.3 Privacy against Attacks

We show the performance of our method against different training and inference phase attacks. If the privacy is fully preserved, the inference attacker has no choice but to random guess the value of the private attribute, the success rate of which is the proportion of the most common value at best. We refer to such a success rate as 'random' and mark them in italics in Tab. 4 and 5. Due to the variety of utility-privacy tradeoffs, we report the results of the best privacy performance and within $1-2\%$ utility decline in each method in the two tables.

Tab. 4 shows the results of CelebA. The line of 'No defense' suggests that all three attacks (basic attack at the inference phase, DSA, and DPA at the training phase) achieve high attack accuracy. Our method significantly lowers the attack accuracy, drawing them close to the accuracy of random guesses, while remaining utility as high as that of no defense. It shows that our method greatly preserves user privacy against all three attacks yet without disrupting the original tasks in both inference and training stages (**Q1& Q2**). Note that adversarial training-based methods are not very effective against black-box attacks in the inference stage. Also formulated by information theoretical loss, the privacy objective for Infocensor seems to be well satisfied at the inference stage, but is below par in facing the training stage attacks, showing that Infocensor's strategy hardly meets privacy-preserving training. A similar conclusion holds for tabular data in Tab. 5 that, our method has superior utility and privacy across all methods, especially with a constantly low attack accuracy during training.

Table 4. Utility and privacy results (%) on CelebA. The best results are marked in bold. 'Basic''ADV''ADV2' refer to the basic black-box attack at the inference stage, adversarial training, and adversarial training with label flip, respectively. The ideal privacy is represented by 'random guesses' in italics.

celebA	**Male(*58.32*)**				**Attractive(*51.25*)**				**WearingLipstick(*52.76*)**			
	Utility	Basic	DSA	DPA	Utility	Basic	DSA	DPA	Utility	Basic	DSA	DPA
No defense	92.27	93.41	84.42	90.42	92.27	79.12	74.83	77.17	92.27	89.91	83.65	88.18
ADV	91.68	89.98	64.39	61.25	91.46	76.91	57.42	61.78	91.69	86.25	58.65	65.42
ADV2	90.89	81.13	59.09	60.12	91.15	71.21	55.62	61.43	91.05	76.88	58.22	64.71
Infocensor	91.37	62.59	58.45	61.79	91.17	62.95	54.27	63.5	91.46	**63.08**	53.09	63.14
Ours	91.29	**61.42**	**58.42**	**58.17**	91.22	**61.53**	**53.26**	**58.18**	91.41	63.14	**52.68**	**61.09**

Table 5. Utility and privacy results (%) on tabular datasets. The best results are marked in bold. The ideal privacy is represented by 'random guesses' in italics.

Health Heritage	**Age(*16*)**			**Gender(*54.23*)**		
	Utility	Basic	DPA	Utility	Basic	DPA
No defense	82.9	31.02	30.07	82.9	59.68	58.39
ADV	82.28	30.03	26.13	82.28	58.66	56.23
ADV2	81.47	28.09	25.58	81.36	56.28	55.88
Infocensor	81.27	**22.86**	24.54	81.2	54.86	55.16
Ours	81.36	23.02	**23.16**	81.12	**54.53**	**55.07**
Adult	**Sex(*66.7*)**			**Relationship(*40*)**		
	Utility	Basic	DPA	Utility	Basic	DPA
No defense	82.68	80.12	77.45	82.68	73.31	67.43
ADV	82.07	78.03	69.39	81.86	71.18	58.26
ADV2	81.72	74.65	68.81	81.75	66.89	57.2
Infocensor	81.94	**66.2**	68.52	81.48	51.98	51.59
Ours	81.95	66.48	**66.39**	81.57	**50.84**	**49.64**

6.4 Privacy-Utility Tradeoffs

For fair and integral comparison, we report results on CelebA under a variety of utility-privacy tradeoffs by tuning the hyperparameters in our method as well as the baselines. As we increase the weight of the privacy task or lower the privacy threshold, data privacy improves but utility declines. The results are visualized in Fig. 4 where the ideal utility-privacy tradeoff is at the right-bottom corner.

As Fig. 4 shows, our method constantly gives a better tradeoff than existing approaches against all attacks (**Q3**), which indicates that our method is able to protect split learning in both training and inference stages. For inference stage attacks, the advantage is the most significant against adversarial training-based methods. For example, in protecting the private attribute 'Male', at a similar utility performance, our method reduces the attack accuracy by over 20%. This is because adversarial training hardly achieves the equilibrium in the attacker-defender game, and thus does not defend adaptive attack strategies. Adversarial training with label flip improves this by obfuscating labels, which drops the attack accuracy by over 5%, but is still far from enough. The performance of Infocensor is marginally inferior to ours in (a)(d) against attacks at the inference phase, but the gap becomes larger in the rest in face of attacks at the training.

Particularly for DPA, the stronger training-stage attack, our method has a clear advantage over all baselines, showing the power of rigorous trajectory control over training. Above all, our method delivers a more desirable tradeoff between utility and privacy, staying closer to the right-bottom corner.

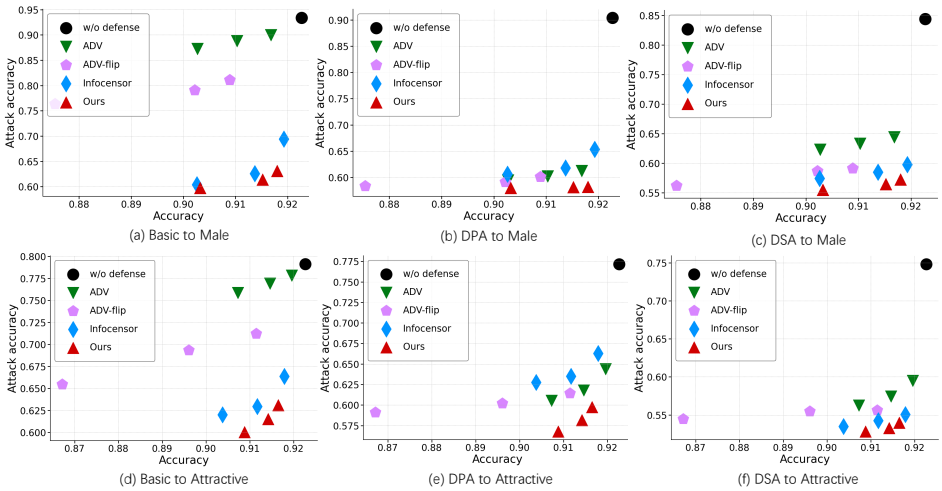

Fig. 4. The privacy-utility tradeoffs on CelebA.

7 Conclusion

We propose a privacy-preserving split learning framework following the paradigm of multi-objective optimization. Different from the conventional linear scalarization in previous works, we seek a Pareto optimal solution to the conflicting goals of utility and privacy while strictly curbing the privacy leakage in training. Our algorithm is essentially a gradient-based, trajectory-controlled Pareto optimal search. Evaluation on different data shows the superior privacy performance of our method against attribute inference attacks both in the training and inference stages, yet without utility decline.

Acknowledgements This work was supported in part by NSF China (62272306, 62136006, 62032020), and a specialized technology project for the pre-research of generic information system equipment (31511130302).

References

1. Doersch, C.: Tutorial on variational autoencoders. arXiv preprint arXiv:1606.05908 (2016)
2. Duong, T., Hazelton, M.L.: Convergence rates for unconstrained bandwidth matrix selectors in multivariate kernel density estimation. Journal of Multivariate Analysis **93**(2), 417–433 (2005)
3. Gupta, O., Raskar, R.: Distributed learning of deep neural network over multiple agents. Journal of Network and Computer Applications **116**, 1–8 (2018)
4. Hershey, J.R., Olsen, P.A.: Approximating the kullback leibler divergence between gaussian mixture models. In: 2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07. vol. 4, pp. IV–317. IEEE (2007)
5. Hillermeier, C.: Nonlinear multiobjective optimization: a generalized homotopy approach, vol. 135. Springer Science & Business Media (2001)
6. Jia, J., Gong, N.Z.: {AttriGuard}: A practical defense against attribute inference attacks via adversarial machine learning. In: 27th USENIX Security Symposium (USENIX Security 18). pp. 513–529 (2018)
7. Kosinski, M., Stillwell, D., Graepel, T.: Private traits and attributes are predictable from digital records of human behavior. Proceedings of the national academy of sciences **110**(15), 5802–5805 (2013)
8. Li, A., Duan, Y., Yang, H., Chen, Y., Yang, J.: TIPRDC: task-independent privacy-respecting data crowdsourcing framework for deep learning with anonymized intermediate representations. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 824–832 (2020)
9. Liu, S., Du, J., Shrivastava, A., Zhong, L.: Privacy adversarial network: representation learning for mobile data privacy. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies **3**(4), 1–18 (2019)
10. Mahapatra, D., Rajan, V.: Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In: International Conference on Machine Learning. pp. 6597–6607. PMLR (2020)
11. Mahapatra, D., Rajan, V.: Exact pareto optimal search for multi-task learning: touring the pareto front. arXiv preprint arXiv:2108.00597 (2021)
12. Moyer, D., Gao, S., Brekelmans, R., Galstyan, A., Ver Steeg, G.: Invariant representations without adversarial training. Advances in Neural Information Processing Systems **31** (2018)
13. Osia, S.A., Taheri, A., Shamsabadi, A.S., Katevas, K., Haddadi, H., Rabiee, H.R.: Deep private-feature extraction. IEEE Transactions on Knowledge and Data Engineering **32**(1), 54–66 (2018)
14. Pasquini, D., Ateniese, G., Bernaschi, M.: Unleashing the tiger: Inference attacks on split learning. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 2113–2129 (2021)
15. Pittaluga, F., Koppal, S., Chakrabarti, A.: Learning privacy preserving encodings through adversarial training. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 791–799. IEEE (2019)
16. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (1978)
17. Salamatian, S., Zhang, A., du Pin Calmon, F., Bhamidipati, S., Fawaz, N., Kveton, B., Oliveira, P., Taft, N.: Managing your private and public data: Bringing down inference attacks against your privacy. IEEE Journal of Selected Topics in Signal Processing **9**(7), 1240–1255 (2015)

18. Sener, O., Koltun, V.: Multi-task learning as multi-objective optimization. Advances in neural information processing systems **31** (2018)
19. Song, C., Shmatikov, V.: Overlearning reveals sensitive attributes. In: 8th International Conference on Learning Representations, ICLR 2020 (2020)
20. Weinsberg, U., Bhagat, S., Ioannidis, S., Taft, N.: Blurme: Inferring and obfuscating user gender based on ratings. In: Proceedings of the sixth ACM conference on Recommender systems. pp. 195–202 (2012)
21. Xiaochen, Z.: Feature Inference Attacks on Split Learning with an Honest-but-Curious Server. Ph.D. thesis, National University of Singapore (2022)
22. Xie, Q., Dai, Z., Du, Y., Hovy, E., Neubig, G.: Controllable invariance through adversarial feature learning. Advances in neural information processing systems **30** (2017)
23. Yang, T.Y., Brinton, C., Mittal, P., Chiang, M., Lan, A.: Learning informative and private representations via generative adversarial networks. In: 2018 IEEE International Conference on Big Data (Big Data). pp. 1534–1543. IEEE (2018)
24. Yao, A.C.: Protocols for secure computations. In: 23rd annual symposium on foundations of computer science (sfcs 1982). pp. 160–164. IEEE (1982)
25. Zheng, T., Li, B.: Infocensor: An information-theoretic framework against sensitive attribute inference and demographic disparity. In: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. pp. 437–451 (2022)

A Notations

We list all notations in this paper in Tab. 6 for better understanding.

B Proof for Loss Formulation

Proof. **Lower bound of $\mathbf{I}(\mathbf{z};\mathbf{y})$.** According to [13], for any conditional distribution $q(y|z)$, it holds that

$$I(z;y) \geq H(y) + \mathbb{E}_{z,y} \log q(y|z). \tag{13}$$

Hence the lower bound of $I(z;y)$ can be defined as

$$\mathcal{L} = H(y) + \max_{\phi} \mathbb{E}_{z,y} \log q_\phi(y|z). \tag{14}$$

The model with parameter ϕ is exactly the classifier performing the original task using uploaded feature z. For fixed z, the better the classifier is trained, the better estimation result it will achieve. Since $H(y)$ is a constant, we only need to optimize the second term of $\mathcal{L}$, which can be translated into minimizing the cross entropy loss. Denoting the cross entropy loss as $CE(\cdot)$, the utility loss can be defined as:

$$L_u = CE(g(f(x)), y).$$

Upper bound of $\mathbf{I}(\mathbf{z};\mathbf{s})$. Assuming s is a discrete variable, the mutual information can be written as

$$I(z;s) = \sum_a p(s_a) \int p(z|s_a) \log \frac{p(z|s_a)}{\sum_b p(s_b) p(z|s_b)} \mathrm{d}z. \tag{15}$$

Table 6. Notations used in this paper

Notations	Description
m	number of tasks
z	intermediate feature
s	sensitive attribute
f_θ	client side model
g_ϕ	server side model
h_ψ	attacker model
L_u	utility loss
L_p	privacy loss
$\mathcal{P}$	Pareto optimal set
$\boldsymbol{r}$	preference vector
G	gradient matrix
C	G^TG
$\boldsymbol{\beta}$	linear combination coefficient of the gradients
$\boldsymbol{d}$	optimization direction
$\boldsymbol{l}$	loss vector
μ_r	non-uniformity
$\boldsymbol{a}$	anchor direction
$Col(\cdot)$	column space
$Null(\cdot)$	null space
$p(\cdot)$	probability distribution

By Jensen's Inequality, $I(z;s)$ is upper bounded by:

$$\mathcal{U} = \sum_a \sum_{b \neq a} p(s_a)p(s_b)KL[p(z|s_a)||p(z|s_b)]. \tag{16}$$

C Proof of Lemma 2

Proof. We prove by contradiction. The formula in (10) can be written as

$$||C\boldsymbol{\beta} - \boldsymbol{a}||_2^2 = ||G^T\boldsymbol{d}||_2^2 + ||\boldsymbol{a}||_2^2 - 2\boldsymbol{a}^TG^T\boldsymbol{d}. \tag{17}$$

If $\boldsymbol{a}^TG^T\boldsymbol{d} < 0$ at convergence, then $||C\boldsymbol{\beta} - \boldsymbol{a}||_2^2 > ||\boldsymbol{a}||_2^2$, which is greater than the result at $\boldsymbol{d} = \boldsymbol{0}$. Hence it contradicts with the fact that the current solution is optimal. Thus $\boldsymbol{a}^TG^T\boldsymbol{d} \geq 0$ holds by solving (10).

D Proof of Lemma 3

Proof. When $\text{rank}(C) = \text{rank}(G) = m$, we can always find $\boldsymbol{\beta}^*$ such that $C\boldsymbol{\beta}^* = \boldsymbol{l}$. Due to the range constraint of $\boldsymbol{\beta}$, we need to scale $\boldsymbol{\beta}^*$ to the range of $[-1, 1]^m$ without altering the direction of $C\boldsymbol{\beta}^*$. Hence we could get $G^T\boldsymbol{d} = C\boldsymbol{\beta} = k\boldsymbol{l} \succ \boldsymbol{0}(k > 0)$, and $\boldsymbol{a}^TG^T\boldsymbol{d} = k\boldsymbol{a}^T\boldsymbol{l} = 0$. The last equality holds due to our claim that $\boldsymbol{a}^T\boldsymbol{l} = 0$.

E Proof of Thm. 1

Proof. If θ is a Pareto optimal solution in optimizing (12), by Pareto Criticality ([5], ch4), G has rank $m-1$. Hence $\text{rank}(C) = \text{rank}(G) = m-1$. Note that $\boldsymbol{a} \neq \mathbf{0}$ at this point, otherwise the optimization terminates for $\boldsymbol{d} = \mathbf{0}$.

Let $\mathcal{S} = \{C\boldsymbol{\beta} | \boldsymbol{\beta} \in [-1,1]^m\}$, $Col(C)$ be the column space of C, and $Null(C)$ be the null space of C. It is clear that $\mathcal{S} \subseteq Col(C)$, and $Col(C)$ and $Null(C)$ are orthogonal complement spaces. By minimizing the objective of (12), $C\boldsymbol{\beta}$ turns out to be the best approximation of $\boldsymbol{a}$ on $\mathcal{S}$. Now we prove by contradiction. Assuming $\boldsymbol{d} = \mathbf{0}$, we have $C\boldsymbol{\beta} = G^T\boldsymbol{d} = \mathbf{0}$. Thus the orthogonal projection of $\boldsymbol{a}$ on $\mathcal{S}$ is $\mathbf{0}$. Hence the orthogonal projection of $\boldsymbol{a}$ on $Col(C)$ is $\mathbf{0}$, which means $\boldsymbol{a} \in Null(C)$. Due to the orthogonality of $\boldsymbol{a}$ and $\boldsymbol{l}$, $\boldsymbol{l} \in Col(C)$, which means $\exists \boldsymbol{\alpha}$ s.t. $C\boldsymbol{\alpha} = G^T G\boldsymbol{\alpha} = \boldsymbol{l} \succeq \mathbf{0}$. So we can find a direction $\boldsymbol{d} = G\boldsymbol{\alpha}$ to decrease all losses, which is contradictory to the condition of Pareto optimality. Therefore, $\boldsymbol{d}$ is not $\mathbf{0}$.

According to Lemma 1, Lemma 2 and Lemma 3, the direction $\boldsymbol{d}$ found by (12) always satisfies $\boldsymbol{a}^T G^T \boldsymbol{d} \geq 0$, so with proper step size, μ keeps decreasing.