Privacy-preserving Federated Adversarial Domain Adaptation over Feature Groups for Interpretability

Yan Kang[1*] , **Yang Liu**[1] , **Yuezhou Wu**[2] and **Guoqiang Ma**[1] and **Qiang Yang**[1,3]

[1]AI Group, WeBank Co., Ltd, Shenzhen, China

[2]School of Computer Science and Engineering, Sun Yat-sen University

[3]Department of Computer Science and Engineering, Hong Kong University of Science and Technology

{yangkang, yangliu, zotrseeewma}@webank.com, wuyzh26@mail2.sysu.edu.cn, qyang@cse.ust.hk

Abstract

We present a novel privacy-preserving federated adversarial domain adaptation approach (**PrADA**) to address an under-studied but practical cross-silo federated domain adaptation problem, in which the party of the target domain is insufficient in both samples and features. We address the lack-of-feature issue by extending the feature space through vertical federated learning with a feature-rich party and tackle the sample-scarce issue by performing adversarial domain adaptation from the sample-rich source party to the target party. In this work, we focus on financial applications where interpretability is critical. However, existing adversarial domain adaptation methods typically apply a single feature extractor to learn feature representations that are low-interpretable with respect to the target task. To improve interpretability, we exploit domain expertise to split the feature space into multiple groups that each holds relevant features, and we learn a semantically meaningful high-order feature from each feature group. In addition, we apply a feature extractor (along with a domain discriminator) for each feature group to enable a fine-grained domain adaptation. We design a secure protocol that enables performing the PrADA in a secure and efficient manner. We evaluate our approach on two tabular datasets. Experiments demonstrate both the effectiveness and practicality of our approach.

1 Introduction

Domain adaptation (DA) methods [Ganin and Lempitsky, 2015; Tzeng *et al.*, 2017; Saito *et al.*, 2018; Wang *et al.*, 2019] has shown notable success. Those methods typically establish alignment or minimize the discrepancy between source and target domains by creating domain-invariant feature representation through deep neural network (DNN) based feature extractors. One major enabler of the adoption of DNN in DA is the availability of a large amount of data with rich features that supports the representation learning of DNN.

Due to increasingly strict legal and regulatory constraints enforced on user privacy, private data from different organizations (domains) cannot be directly integrated for training machine learning models. In recent years, federated learning (FL) has emerged as a practicable solution to tackle data silo issues without compromising user privacy.

Recently, a growing number of works have been proposed to integrate domain adaptation into *cross-silo* FL setting [Peterson, 2019; Peng *et al.*, 2019; Li *et al.*, 2020; Song *et al.*, 2020] for mitigating domain shift among independent parties. These works conduct experiments typically using image or text data that have sufficient features to perform meaningful representation learning. However, in many real-world FL applications (e.g., finance and retail) where data is stored in tabular format, the participating parties might have insufficient features for building DNN-based domain adaptation models. In addition, mainstream adversarial DA methods [Ganin and Lempitsky, 2015; Tzeng *et al.*, 2017; Wang *et al.*, 2019] typically apply a single feature extractor over the whole feature space to learn transferable feature representations, which are likely to be high-dimensional and low-interpretable with respect to the target task.

In this work, we focus on financial applications where the target party has insufficient samples and features, and the model interpretability is an important concern. To address the lack-of-feature issue, we propose to extend the feature space by collaborating with a feature-rich party through *vertical (feature-partitioned) federated learning (VFL)* [Yang *et al.*, 2019]. We propose to split the feature space into groups of relevant features and apply fine-grained DA to each group to improve both feature transferability and model interpretability. To protect data privacy, we develop a VFL framework that enables collaborative domain adaptation modeling and utilizes Partially Homomorphic Encryption (PHE) [Paillier, 1999] to protect private data.

The contributions of this paper are highlighted as follows:

1. To the best of our knowledge, this work is the first attempt to study adversarial domain adaptation in the VFL setting based on tabular data.

2. We develop a privacy-preserving VFL framework that allows participating parties to collaboratively conduct domain adaptation without exposing private data.

3. We propose a fine-grained domain adaptation over feature groups to reduce feature dimensionality, enhance model interpretability, and facilitate the learning of domain-invariant features.

---

*Contact Author

2 Related Work

2.1 Federated Domain adaptation

Federated domain adaptation aims to conduct domain adaptation modeling among independent parties of different domains without violating privacy. [Peterson, 2019] applies a mixture of experts (MoE) strategy that each participant combines a collaboratively-learned general model and a domain-tuned private model to reconcile distribution differences among participants. [Peng *et al.*, 2019] leverages federated adversarial domain alignment with a dynamic attention mechanism to enhance knowledge transfer. [Mohri *et al.*, 2019] proposes agnostic federated learning aiming to optimize the global model for any target distribution formed by a mixture of client distributions without overfitting data of any particular client.

2.2 Deep Neural Network on Encrypted Data

Protecting privacy is a crucial element of federated learning. Homomorphic encryption (HE) is one of the major solutions to address the privacy issue. Although HE is a promising solution that allows computation to be performed on encrypted data, its expensive computational cost makes it impractical to be applied in training an entire DNN model. To address this issue, GELU-NET [Zhang *et al.*, 2018] adopts a client-server architecture in which the client encrypts the data while the server performs most computation on encrypted data. ACML [Zhang and Zhu, 2020] focuses on a federated learning scenario where data and labels are distributed among two independent parties that each owns one part of a DNN model. It proposes to perform costly encryption-decryption operations only on the boundary of the two partial DNN models, leaving the rest computation conducted in plaintext.

2.3 Model Interpretability

Many methods have been proposed for interpreting deep neural networks [Montavon *et al.*, 2018]. These methods focus on post-hoc interpretability that analyzes the relationships between input and output of the trained model rather than elucidating model's internal structures. Other methods [Chen *et al.*, 2019; Ho, 2020; Alvarez-Melis and Jaakkola, 2018] construct prototypes or general concepts that shed light on the decision-making process. e.g., [Chen *et al.*, 2019] proposed ProtoPNet that learns a set of prototypes each can be considered as the latent representation of a small prototypical part of training images. Then, the label prediction can be made based on a weighted combination of the similarity scores between parts of the image and the learned prototypes.

3 Problem Definition

We consider following cross-silo federated domain adaptation scenario that involves three parties. Party A is from the target domain and it has only a small number of samples $(\mathbf{X}^A, \mathbf{Y}^A) \in \mathbb{R}^{n^A \times (m+1)}$. Party B is from the source domain and it has a large amount of samples $(\mathbf{X}^B, \mathbf{Y}^B) \in \mathbb{R}^{n^B \times (m+1)}$. These two parties share the same feature space and have the same task. We consider conduct domain adaptation from party B to party A, and we call these two parties *ac-*

tive parties because they initiate the domain adaptation procedure. The two active parties have insufficient number of features to support domain adaptation. Thus, we refer to a *passive party* C that is able to provide sufficient amount of complementary features $\mathbf{X}^{B^c} \in \mathbb{R}^{n^B \times m^c}$ and $\mathbf{X}^{A^c} \in \mathbb{R}^{n^A \times m^c}$ for party B and party A, respectively. $\mathbf{X}^{B^c}$ and $\mathbf{X}^{A^c}$ have the same feature space, and $n^B \gg n^A$ and $m^c \gg m$.
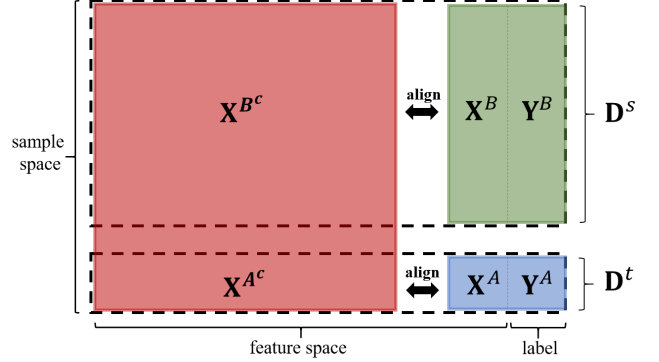

Figure 1: The virtual tabular data of the cross-silo federated domain adaptation. Party A has a small amount of samples $(\mathbf{X}^A, \mathbf{Y}^A)$ while Party B has a large amount of samples $(\mathbf{X}^B, \mathbf{Y}^B)$. Party C provides complementary features $\mathbf{X}^{B^c}$ for party B and $\mathbf{X}^{A^c}$ for party A. Thus, we form virtual datasets $\mathbf{D}^s = [\mathbf{X}^{B^c}; \mathbf{X}^B; \mathbf{Y}^B]$ of the source domain and $\mathbf{D}^t = [\mathbf{X}^{A^c}; \mathbf{X}^A; \mathbf{Y}^A]$ of the target domain.

We align $\mathbf{X}^{B^c}$ with $(\mathbf{X}^B, \mathbf{Y}^B)$ along the feature axis to form a virtual dataset $\mathbf{D}^s = [\mathbf{X}^{B^c}; \mathbf{X}^B; \mathbf{Y}^B]$ of the source domain. Likewise, we form a virtual dataset $\mathbf{D}^t = [\mathbf{X}^{A^c}; \mathbf{X}^A; \mathbf{Y}^A]$ of the target domain. The alignment can be performed by leveraging privacy-preserving entity matching approaches [Hardy *et al.*, 2017]. Figure 1 shows virtual tabular datasets $\mathbf{D}^s$ and $\mathbf{D}^t$ formed among the three parties.

Under this setting, our PrADA approach is conducted from two aspects: (1) extending the feature space of active parties A and B through vertical federated learning with a feature-rich passive party C; (2) performing domain adaptation from party B of the sample-rich source domain to party A of the sample-scarce target domain based on the extended yet distributed feature space. *Our ultimate goal is to improve the performance of the target task of party A.*

4 Architecture Overview

Figure 2 illustrates the architecture of PrADA. Active party $p \in \{A, B\}$ owns label predictors R^p while the passive party C owns feature extractors $\mathscr{F} = \{F_i\}_{i=1}^k$ and domain discriminators $\mathscr{D} = \{D_i\}_{i=1}^k$ for extracting transferable feature representations from multiple feature groups. Instead of directly passing feature representations to the active party for training the label predictor, party C leverages a set of aggregators $\mathscr{G} = \{G_i\}_{i=1}^k$ to compress each feature representation learned from a feature group into a scalar value representing a high-order feature and feeds these high-order features into the label predictor. Active parties have no feature extractors because we assume active parties have insufficient raw features to conduct effective domain adaptation, and these raw

features are highly business-related such that their contributions to the model decision should be interpretable. Thus, active parties feed their raw features directly into their label predictors. The feature extractors and domain discriminators are trained locally at party C while label predictor is trained collaboratively by both active party (either A or B) and passive party C in a privacy-preserving manner, which we will elaborate in the section 7.
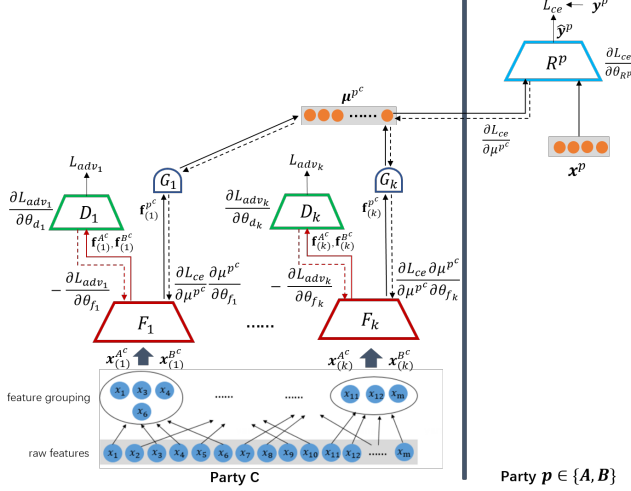

Figure 2: Architecture of PrADA. Passive party C locally trains k pairs of $\{F_i, D_i\}_{i=1}^{k}$ that each corresponds to a feature group by optimizing (4). Active party $p \in \{A, B\}$ and passive party C collaboratively train $\{F_i\}_{i=1}^{k}$ and R^p by optimizing (5) using μ^{p^c}, $\mathbf{x}^p$ and y^p.

5   Feature Grouping

PrADA leverages feature grouping (FG) to improve both the interpretabiltiy of label predictors and the transferability of feature extractors. We propose that party C creates multiple feature groups from the feature space such that features in the same group are tightly relevant, and party C assigns each feature group a feature extractor alone with a domain discriminator to learn domain-invariant feature representations. We hypothesize that a more fine-grained adaptation between each of the two domains' feature group can improve the transferability of the domain-invariant feature representations.

We adopt logistic regression (LR) model as the label predictor. LR considers each input feature as a fundamental interpretable unit. Therefore, instead of directly passing feature representations learned from feature extractors to LR model of active party, party C leverages a set of aggregators to compress each feature representation into a scalar value representing a high-order feature, and then it feeds these high-order features into the LR model. As a result, the LR model takes as input a manageable number of meaningful high-order features (from party C). The procedure of party C generating high-order features is described as follows:

(1) Party C turns to domain expertise to group its features into k feature groups. Based on this grouping, we obtain k groups of relevant features $\{\mathbf{x}_{(i)}^{p^c}\}_{i=1}^{k}$ for each sample

$\mathbf{x}^{p^c} \in \mathbb{R}^{1 \times m^c}$ dawn from $\mathbf{X}^{p^c}, p \in \{A, B\}$. (2) Party C leverages k feature extractors $\mathscr{F} = \{F_i\}_{i=1}^{k}$ to extract k feature representations $\{\mathbf{f}_{(i)}^{p^c}\}_{i=1}^{k} = \{F_i(\mathbf{x}_{(i)}^{p^c})\}_{i=1}^{k}$. (3) Party C leverages k aggregators $\{G_i\}_{i=1}^{k}$ to compress $\{\mathbf{f}_{(i)}^{p^c}\}_{i=1}^{k}$ into a vector of k high-order feature values:

$$\boldsymbol{\mu}^{p^c} = [G_1(\mathbf{f}_{(1)}^{p^c}); G_2(\mathbf{f}_{(2)}^{p^c}); \ldots ; G_k(\mathbf{f}_{(k)}^{p^c})] \qquad (1)$$

where $G_k(\mathbf{f}_{(k)}^{p^c})$ returns a scalar value representing the high order feature for feature group $\mathbf{x}_{(i)}^{p^c}$. Figure 2 depicts the procedure described above. For performing federated adversarial domain adaptation, party C feeds $\{\mathbf{f}_{(i)}^{p^c}\}_{i=1}^{k}$ into domain discriminators for minimizing domain discrimination losses and passes $\boldsymbol{\mu}^{p^c}$ to active party for minimizing label prediction loss, as described in section 6.

6   Federated Adversarial Domain adaptation

The federated adversarial domain adaptation of PrADA involves two stages: *pre-training* and *fine-tuning*. The pre-training stage is performed between the source party B and party C. It aims to train feature extractors that can learn both domain-invariant and label-discriminative features. The fine-tuning stage is performed between the target party A and party C, and it aims to train the target LR model of party A leveraging pre-trained feature extractors.

6.1   Pre-training Stage

The essential idea of adversarial domain adaptation is to train feature extractors to learn features that are both discriminative to the task and invariant to the change of domains.

In our setting, party C leverages k feature extractors $\mathscr{F} = \{F_i\}_{i=1}^{k}$ and their corresponding k domain discriminators $\mathscr{D} = \{D_i\}_{i=1}^{k}$ to learn domain-invariant feature representations from k feature groups. Specifically, the ith feature extractor F_i learns feature representation from the ith feature group and then the ith domain discriminator D_i maps this feature representation to a domain label $d \in \{0, 1\}$. The overall domain discrimination loss is computed as follows:

$$L_{adv}(\mathscr{F}, \mathscr{D}) = -\mathbb{E}_{\mathbf{x}^{B^c} \sim \mathbf{X}^{B^c}} \sum_{i=1}^{k} log[D_i(F_i(\mathbf{x}_{(i)}^{B^c}))]$$
$$- \mathbb{E}_{\mathbf{x}^{A^c} \sim \mathbf{X}^{A^c}} \sum_{i=1}^{k} log[1 - D_i(F_i(\mathbf{x}_{(i)}^{A^c}))] \quad (2)$$

To make feature extractors obtain task-specific discriminative features, we optimize label prediction loss to train both source LR model R^B of party B and feature extractors $\mathscr{F}$ via VFL between party B and party C. The label prediction loss is defined as follows:

$$L_{ce}(\mathscr{F}, R^B) =$$
$$\mathbb{E}_{(\mathbf{x}^{B^c}, \mathbf{x}^B, \mathbf{y}^B) \sim \mathbf{D}^B}[\ell_{ce}(R^B([\boldsymbol{\mu}^{B^c}; \mathbf{x}^B]), \mathbf{y}^B)] \quad (3)$$

where $\boldsymbol{\mu}^{B^c}$ is the high-order feature vectors passed from party C and $\mathbf{x}^B$ is the feature vectors possessed by party B.

In our federated setting, L_{adv} and L_{ce} are optimized separately: $L_{adv}(\mathscr{F}, \mathscr{D})$ is optimized locally at party C since it only involves data of party C, while $L_{ce}(\mathscr{F}, R^B)$ is optimized collaboratively by party B and party C through VFL since it involves features of the two parties. To this end, we train parameters $\{\theta_{f_i}\}_{i=1}^k$ of feature extractors $\mathscr{F}$, $\{\theta_{d_i}\}_{i=1}^k$ of domain discriminators $\mathscr{D}$, and θ_{R^B} of LR model R^B by solving following two optimization problems.

$$\underset{\{\theta_{f_i}\}_{i=1}^k}{\operatorname{argmin}} \underset{\{\theta_{d_i}\}_{i=1}^k}{\operatorname{argmax}} (-\lambda L_{adv}(\mathscr{F}, \mathscr{D})) \qquad (4)$$

$$\underset{\{\theta_{f_i}\}_{i=1}^k, \theta_{R^B}}{\operatorname{argmin}} L_{ce}(\mathscr{F}, R^B) \qquad (5)$$

where λ is a hyperparameter that controls the trade-off between the two losses $\{\theta_{d_i}\}_{i=1}^k$ are trained by minimizing the domain discrimination loss, θ_{R^B} is trained by minimizing the label prediction loss, and $\{\theta_{f_i}\}_{i=1}^k$ are trained by minimizing the label prediction loss while at the same time maximizing the domain discrimination loss.

Figure 2 shows the workflow of pre-training stage and Algorithm 1 describes the procedure of optimizing (4) and (5).

Algorithm 1 Federated Pre-training

1: **Initialization**: feature extractors $\mathscr{F}$, domain discriminators $\mathscr{D}$, batch indices $\mathcal{I}$
2: **Input**: $\mathbf{D}^s = [\mathbf{X}^{B^c}; \mathbf{X}^B; \mathbf{Y}^B], \mathbf{X}^{A^c}$
3: **for** $e = 1, 2, ..., E$ **do**
4:     **for** $i \in \mathcal{I}$ **do**
5:         **Party C** do:
6:             $\mathbf{x}^{A^c} \leftarrow$ sample a mini-batch from $\mathbf{X}^{A^c}$;
7:             $\mathbf{x}^{B^c} \leftarrow$ select ith mini-batch from $\mathbf{X}^{B^c}$;
8:             update models in $\mathscr{F}, \mathscr{D}$ by optimizing (4) using $\mathbf{x}^{B^c}$ and $\mathbf{x}^{A^c}$;
9:             compute $\boldsymbol{\mu}^{B^c}$ by (1) using $\mathbf{x}^{B^c}$;
10:            encrypt $\boldsymbol{\mu}^{B^c}$ and send $[[\boldsymbol{\mu}^{B^c}]]$ to party B;
11:         **Party B** do:
12:            $\mathbf{x}^B \leftarrow$ select ith min-batch from $\mathbf{X}^B$;
13:            $\mathbf{y}^B \leftarrow$ select ith min-batch from $\mathbf{Y}^B$;
14:         **Party B** and **Party C** do:
15:            minimize (5) using Algorithm 2 with $[[\boldsymbol{\mu}^{B^c}]], \mathbf{x}^B, \mathbf{y}^B$;
16:     **end for**
17: **end for**

6.2 Fine-tuning Stage

The fine-tuning stage aims to train LR model R^A of party A using target data $\mathbf{D}^t$ based on pre-trained feature extractors of party C. For each iteration, party C applies (1) to compute feature vectors $\boldsymbol{\mu}^{A^c}$ and then send homomorphically encrypted $[[\boldsymbol{\mu}^{A^c}]]$ to party A for computing the label prediction loss:

$$L_{ce}(\mathscr{F}, R^A) =$$
$$\mathbb{E}_{(\mathbf{x}^{A^c}, \mathbf{x}^A, \mathbf{y}^A) \sim \mathbf{D}^A}[\ell_{ce}(R^A([\boldsymbol{\mu}^{A^c}; \mathbf{x}^A]), \mathbf{y}^A)] \quad (6)$$

The fine-tuning stage follows similar procedure described in Algorithm 1 except that it does not require party C to optimize (4). Since parties B and A are two independent parties,

---

the source LR model R^B cannot be reused by party A and thus the target LR model R^A *has to be trained from the random initialization.*

Algorithm 2 Privacy-preserving Federated Training

1: **Input**: $[[\boldsymbol{\mu}^C]], \mathbf{x}^p, \mathbf{y}^p$, where $p \in \{B, A\}$
2:     run Algorithm 3 with $[[\boldsymbol{\mu}^C]], \mathbf{x}^p, \mathbf{y}^p$;
3:     run Algorithm 4 with $[[\boldsymbol{\mu}^C]], \mathbf{x}^p$;

---

7 Privacy-preserving Federated Learning

As shown in (3) (6), minimizing label prediction loss for training LR model involves data from a active party (either party B or A) and the passive party C. In this section, we elaborate our proposed secure protocol that enable the two independent parties to collaboratively train the LR model without exposing their data. First, we define the LR model as follows:

$$R^p([\boldsymbol{\mu}^C; \mathbf{x}^p]) = \sigma([\boldsymbol{\mu}^C; \mathbf{x}^p]\mathbf{W} + b) \qquad (7)$$

where σ is the sigmoid function and $p \in \{A, B\}$ denote a active party, $\mathbf{W} \in \mathbb{R}^{m+k}$ is the weights of model R^p and $b \in \mathbb{R}^1$ is the bias. In this section, we denote $\boldsymbol{\mu}^C$ as the feature vectors from party C and $\mathbf{x}^p$ as raw features from the active party p. We further decompose the input of σ as follows:

$$z = \boldsymbol{\mu}^C \mathbf{W}^C + \mathbf{x}^p \mathbf{W}^p + b^p \qquad (8)$$

where $\mathbf{W}^C \in \mathbb{R}^k$ is for the input $\boldsymbol{\mu}^C$ from party C while $\mathbf{W}^p \in \mathbb{R}^m$ is for the input $\mathbf{x}^p$ from party p. Note that both $\mathbf{W}^p$ and $\mathbf{W}^C$ are owned by the active party p, but the real value of $\mathbf{W}^C$ is concealed from party p during training so that party p cannot infer private data of party C.

We extend the PHE-based secure protocol applied to the setting where one party has features and another has only labels [Zhang and Zhu, 2020] to our setting where features are distributed among two parties. Our new secure protocol includes two stages, described in section 7.1 and 7.2, respectively. We denote the PHE encryption, addition and multiplication as $[[\cdot]], \oplus$ and $\otimes$, respectively. Note that in our setting, only party C can encrypt and decrypt exchanging messages.

7.1 Privacy-Preserving Forward Propagation

Algorithm 3 aims to compute the label prediction loss in (3) without exposing private data. To this end, party C encrypts μ^C with PHE and sends encrypted $[[\mu^C]]$ to party p to prevent privacy leakage. When receiving $[[\mu^C]]$, party p computes $[[\tilde{z}^C]]$ and z^p, and sends $[[\tilde{z}^C]]$ to party C with random noise ϵ^p. Party C then decrypts $[[\tilde{z}^C + \epsilon^p]]$, adds it with $\boldsymbol{\mu}^C \varepsilon_t^C$, and sends the result to party p. The $\boldsymbol{\mu}^C \varepsilon_t^C$ is for cancelling out the accumulated random noise ε_t^C embedded in $\widetilde{\mathbf{W}}_t^C$ at current iteration. Last, party p computes the loss $\ell_{ce}(\sigma(z), \mathbf{y}^p)$.

7.2 Privacy-Preserving Backward Propagation

During the privacy-preserving backward propagation as described in Algorithm 4, the active party p securely updates LR model R^p and backpropagate gradients to party C. As shown in (8), we partitioned weights of R^p into $\mathbf{W}^p$ and $\mathbf{W}^C$. On

Algorithm 3 Privacy-preserving Forward Propagation

1: **Initialization**: LR model parameters $\widetilde{\mathbf{W}}_0^C$ and $\mathbf{W}_0^p$, accumulated noise ε_0^C
2: **Input**: $[[\boldsymbol{\mu}^C]], \mathbf{x}^p, \mathbf{y}^p, p \in \{A, B\}$
3: **Party p**:
4:    compute logit:
5:       $[[\tilde{z}^C]] \leftarrow [[\boldsymbol{\mu}^C]] \otimes \widetilde{\mathbf{W}}_t^C$;
6:       $z^p \leftarrow \mathbf{x}^p \mathbf{W}_t^p + b^p$;
7:    add noise $[[\tilde{z}^C + \epsilon^p]] \leftarrow [[\tilde{z}^C]] \oplus \epsilon^p$;
8:    send $[[\tilde{z}^C + \epsilon^p]]$ to party $\boldsymbol{C}$;
9: **Party C**:
10:    $\tilde{z}^C + \epsilon^p \leftarrow$ decrypt $[[\tilde{z}^C + \epsilon^p]]$;
11:    $z^C + \epsilon^p \leftarrow \tilde{z}^C + \boldsymbol{\mu}^C \varepsilon_t^C + \epsilon^p$;
12:    send $z^C + \epsilon^p$ to party $\boldsymbol{p}$;
13: **Party p**:
14:    remove noise $z^C \leftarrow z^C + \epsilon^p$;
15:    $z \leftarrow z^p + z^C$;
16:    compute loss $\ell_{ce}(\sigma(z), \mathbf{y}^p)$;

Algorithm 4 Privacy-preserving Backward Propagation

1: **Initialization**: learning rate η
2: **Input**: $[[\boldsymbol{\mu}^C]], \mathbf{x}^p, p \in \{A, B\}$
3: **Party p**:
4:    $\delta^l \leftarrow \nabla_\sigma \ell_{ce}$ w.r.t the activation function σ;
5:    backpropagate gradients δ^l:
6:       $[[\Delta \mathbf{W}_t^C]] \leftarrow [[\boldsymbol{\mu}^C]] \otimes \delta^l$;
7:       $\Delta \mathbf{W}_t^p \leftarrow \mathbf{x}^p \delta^l$;
8:       $\Delta b_t^p \leftarrow \delta^l$;
9:    add noise $[[\Delta \mathbf{W}_t^C + \epsilon^p]] \leftarrow [[\Delta \mathbf{W}_t^C]] \oplus \epsilon^p$;
10:    send $[[\Delta \mathbf{W}_t^C + \epsilon^p]]$ to party $\boldsymbol{C}$;
11: **Party C**:
12:    $\Delta \mathbf{W}_t^C + \epsilon^p \leftarrow$ decrypt $[[\Delta \mathbf{W}_t^C + \epsilon^p]]$;
13:    add noise $\Delta \widetilde{\mathbf{W}}_t^C + \epsilon^p \leftarrow \Delta \mathbf{W}_t^C + \frac{\epsilon^C}{\eta} + \epsilon^p$;
14:    $\varepsilon_{t+1}^C \leftarrow \varepsilon_t^C + \epsilon^C$ and $[[\varepsilon_{t+1}^C]] \leftarrow$ encrypt ε_{t+1}^C;
15:    send $[[\varepsilon_{t+1}^C]], \Delta \widetilde{\mathbf{W}}_t^C + \epsilon^p$ to party $\boldsymbol{p}$
16: **Party p**:
17:    remove noise $\Delta \widetilde{\mathbf{W}}_t^C \leftarrow \Delta \widetilde{\mathbf{W}}_t^C + \epsilon^p$
18:    update weights and bias of logistic regression model:
19:       $\widetilde{\mathbf{W}}_{t+1}^C \leftarrow \widetilde{\mathbf{W}}_t^C - \eta \Delta \widetilde{\mathbf{W}}_t^C$;
20:       $\mathbf{W}_{t+1}^p \leftarrow \mathbf{W}_t^p - \eta \Delta \mathbf{W}_t^p$;
21:       $b_{t+1}^p \leftarrow b_t^p - \eta \Delta b_t^p$;
22:    $[[\delta^C]] \leftarrow \delta^l \otimes (\widetilde{\mathbf{W}}_{t+1}^C \oplus [[\varepsilon_{t+1}^C]])$;
23:    send $[[\delta^C]]$ to party $\boldsymbol{C}$;
24: **Party C**:
25:    $\delta^C \leftarrow$ decrypt $[[\delta^C]]$;
26:    update feature aggregators in $\mathscr{G}$ and feature extractors
27:    in $\mathscr{F}$ based on gradient δ^C using SGD;

---

one hand, party p updates $\mathbf{W}_t^p$ and b_t^p in plaintext since party p owns these weights. On the other hand, party p collaborates with party C to securely update $\mathbf{W}^C$. Specifically, party p computes encrypted $[[\Delta \mathbf{W}_t^C]]$, adds it with noise ϵ^p and then sends $[[\Delta \mathbf{W}_t^C + \epsilon^p]]$ to party C. Party C in turn decrypts $[[\Delta \mathbf{W}_t^C + \epsilon^p]]$ and sends $\Delta \widetilde{\mathbf{W}}_t^C + \epsilon^p$ back to party p, where $\Delta \widetilde{\mathbf{W}}_t^C = \Delta \mathbf{W}_t^C + \frac{\epsilon^C}{\eta}$, ϵ^C is the random noise added by party C and η is the learning rate. To cancel out the accumulated noise embedded in $\widetilde{\mathbf{W}}_{t+1}^C$, party C sends the encrypted accumulated noise $[[\varepsilon_{t+1}^C]]$ to party p, which in turn calculates the gradient $[[\delta^C]]$ of loss ℓ_{ce} w.r.t μ^C and sends $[[\delta^C]]$ back to party C, which decrypts $[[\delta^C]]$ and backpropagates δ^C locally.

8 Experiments

8.1 Experimental datasets and settings

We evaluate our proposed PrADA based on two datasets: one is Census Income dataset, and another is a real-world financial dataset called PPD loan default. For each dataset, we run experiments under following four settings:

1. **A-Local**: Party A only uses its local data to train models without leveraging VFL and DA.

2. **A-VFL**: Party A uses target domain data $\mathbf{D}^t$ to train models via VFL with party C.

3. **AB-VFL**: Assuming party A and B are from the same organization. Party A uses $\mathbf{D}^t$ and $\mathbf{D}^s$ of both domains to train models via VFL with party C (with no DA).

4. **B $\rightarrow$ A**: We conduct PrADA discussed in sections 5 and 6 to perform federated adversarial domain adaptation from party B to party A.

In settings 2 and 3, we adopt SecureLR and SecureBoost implemented in FATE [WeBank, 2018] as comparing models. These two models are VFL version of XGBoost and LR, respectively, and they are using PHE to protect data privacy. To explore the effectiveness of PrADA model, we propose two

different ablations, i.e. PrADA$_{\text{w/o DA}}$: without domain adaptation; PrADA$_{\text{w/o FG}}$: without feature grouping. We report AUC and KS (Kolmogorov-Smirnov test) [Chakravarti *et al.*, 1967] of all trained models on the *test data of target party A*.

8.2 Experiments on Census Income

Census income is a census dataset from the UCI Machine Learning Repository. We split this dataset into a *undergraduate* source domain and a *postgraduate* target domain. The source domain has 80000 examples while the target domain has 4000 examples. Our goal is to help party A of the target domain to predict whether a person's income exceeds 50,000 US dollars or not.

After data preprocessing, the census income dataset contains 36 features, 31 of which are categorical. We put 5 numerical features on active parties (i.e., A and B) while the 31 categorical features on passive party C. We split 31 features on party C into 4 feature groups including *employment*, *demographics*, *household*, and *migration*. Thus, the LR model has parameter dimension of 9 (5+4).

The experimental results are shown in Table 1. From these, we observe that: (1) The performance of models consistently improves from the **A-Local** to **A-VFL** and to **AB-VFL**, demonstrating that leveraging additional features and samples indeed improve the model performance. (2) PrADA outperforms PrADA$_{\text{w/o DA}}$ of **AB-VFL** setting in AUC by 2.0% and in KS by 2.5%, demonstrating the effectiveness of

Setting	Model	AUC (%)	KS (%)
A-Local	LR	70.65	34.85
	XGBoost	72.29	39.82
A-VFL	SecureLR	71.56	35.21
	SecureBoost	74.90	40.87
	PrADA$_{w/o\ DA}$	73.12	37.18
AB-VFL	SecureLR	73.76	40.16
	SecureBoost	78.23	44.64
	PrADA$_{w/o\ DA}$	78.52±0.40	44.56±0.52
B → A	PrADA$_{w/o\ FG}$	80.01±0.30	46.35±0.24
	PrADA	80.53±0.32	47.06±0.26

Table 1: Comparison between models on census income dataset.

Setting	Model	AUC (%)	KS (%)
A-Local	LR	54.34	11.10
	XGBoost	54.29	9.34
A-VFL	SecureLR	62.97	22.43
	SecureBoost	65.78	24.79
	PrADA$_{w/o\ DA}$	64.85	26.02
AB-VFL	SecureLR	72.55	39.28
	SecureBoost	74.69	41.66
	PrADA$_{w/o\ DA}$	75.03±0.40	41.78±0.41
B → A	PrADA$_{w/o\ FG}$	75.21±0.31	41.04±0.26
	PrADA	75.33±0.18	42.56±0.35

Table 2: Comparison between models on PPD loan default dataset.

PrADA on bridging the divergence between source and target domains. (3) PrADA outperforms PrADA$_{w/o\ FG}$ in AUC and KS by 0.52% and 0.71% respectively, demonstrating the effectiveness of FG-based domain adversarial training on improving the transferability of feature extractors.

8.3 Experiments on Loan Default

PPD loan default is a loan default risk dataset for the online lending industry published by FinVolution Group. It contains loan data issued in 2014. We consider 40000 samples of loans issued in the first three quarters of 2014 as the source domain while the 3000 samples of loans issued in the forth quarter as the target domain. This is an Out-Of-Time scenario in financial risk control. Our goal is to help party A to build a loan predictor to predict whether a loan will default or not.

After data preprocessing, PPD dataset has 162 features, 27 of which are categorical. For protecting privacy, user and feature names are anonymized. We put 6 *demographics* features and labels on active parties while the rest 156 features on passive party C. We split features of party C into 5 groups including *user location*, *third-party period*, *education*, *social network*, and *micro-blog*. We embed all categorical features. The LR model has parameter dimension of 11 (6+5)

The experimental results are reported in Table 2. From these, we observe that: (1) Table 2 reports a similar trend as Table 1 that the performance of models improves with more data involved in training. (2) PrADA outperforms PrADA$_{w/o\ DA}$ of **AB-VFL** setting in AUC by 0.3% and in KS by 0.78%, demonstrating the effectiveness of PrADA on mitigating domain divergence. (3) PrADA outperforms PrADA$_{w/o\ FG}$, demonstrating the superiority of FG-based DA over conventional DA.

8.4 Model Interpretability

We demonstrate model interpretability by visualizing the impact of features on target model R^A using SHAP [Lundberg and Lee, 2017], which is a tool widely used to explain blackbox models. We select census income dataset for this purpose since the semantic meaning of features in PPD is anonymized.

Figure 3 plots the SHAP values of every feature for all samples to illustrate the impact of those features on the prediction output. Features are ranked in descending order. The color represents the feature value (red high, blue low). The horizontal location shows whether the effect of a feature value is
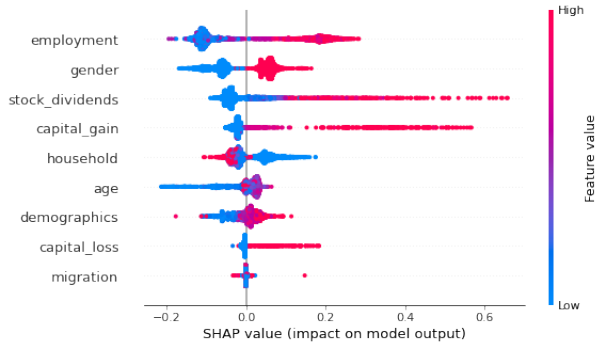

Figure 3: The impact of features on model predictions.

associated with a higher or lower prediction. Specifically, *employment*, *gender*, *stock_dividends*, *captial_gain* and *household* are the five most influential features, in which *household* is negatively correlated with the prediction while the other four features have positive impact on the label prediction.

8.5 Computation Cost

We compare the computation cost among SecureLR, SecureBoost and PrADA using FATE 1.6. The experiments are conducted on a machine with 72 Intel Xeon Gold 6140 CPUs and 320 GB RAM. All experiments are simulated in standalone deployment mode.

Setting	Model	Time(h)	
AB-VFL	SecureLR	1.12	
	SecureBoost	2.16	
	PrADA$_{w/o\ DA}$	4.10	
		PT Time	FT Time
B → A	PrADA	5.47	0.52

Table 3: Training time (hours) on census income dataset. PT and FT denote pre-taining time and fine-tuning time, respectively.

Table 3 reports the results that the training time of PrADA$_{w/o\ DA}$ is 4.10 hours, which is approximately twice the training time spent by SecureBoost. In the setting of **B → A**, PrADA takes 5.47 hours to train because DA involved. However, once pre-train is finished, PrADA only takes half an hour to performance fine-tune stage.

References

[Alvarez-Melis and Jaakkola, 2018] David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks. NIPS'18, page 7786–7795, Red Hook, NY, USA, 2018. Curran Associates Inc.

[Chakravarti *et al.*, 1967] I. M. Chakravarti, R. G. Laha, and J. Roy. *Handbook of Methods of Applied Statistics, Volume I*. John Wiley and Sons, New York, NY, 1967.

[Chen *et al.*, 2019] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: Deep learning for interpretable image recognition. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8930–8941. Curran Associates, Inc., 2019.

[Ganin and Lempitsky, 2015] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1180–1189, Lille, France, 07–09 Jul 2015. PMLR.

[Hardy *et al.*, 2017] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *CoRR*, abs/1711.10677, 2017.

[Ho, 2020] Daniel Ho. Nbdt: Neural-backed decision trees. Master's thesis, EECS Department, University of California, Berkeley, May 2020.

[Li *et al.*, 2020] Xiaoxiao Li, Yufeng Gu, Nicha Dvornek, Lawrence H. Staib, Pamela Ventola, and James S. Duncan. Multi-site fmri analysis using privacy-preserving federated learning and domain adaptation: Abide results. *Medical Image Analysis*, 65:101765, 2020.

[Lundberg and Lee, 2017] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[Mohri *et al.*, 2019] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *36th International Conference on Machine Learning, ICML 2019*, 36th International Conference on Machine Learning, ICML 2019, pages 8114–8124. International Machine Learning Society (IMLS), January 2019. 36th International Conference on Machine Learning, ICML 2019 ; Conference date: 09-06-2019 Through 15-06-2019.

[Montavon *et al.*, 2018] Grégoire Montavon, Wojciech Samek, and Klaus Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing: A Review Journal*, 73:1–15, February 2018.

[Paillier, 1999] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[Peng *et al.*, 2019] Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. *arXiv preprint arXiv:1911.02054*, 2019.

[Peterson, 2019] D. Peterson. Private federated learning with domain adaptation. *ArXiv*, abs/1912.06733, 2019.

[Saito *et al.*, 2018] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[Song *et al.*, 2020] L. Song, C. Ma, G. Zhang, and Y. Zhang. Privacy-preserving unsupervised domain adaptation in federated setting. *IEEE Access*, 8:143233–143240, 2020.

[Tzeng *et al.*, 2017] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2962–2971, 2017.

[Wang *et al.*, 2019] Z. Wang, Z. Dai, B. Póczos, and J. Carbonell. Characterizing and avoiding negative transfer. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11285–11294, 2019.

[WeBank, 2018] WeBank. FATE, a secure computing framework to support the federated ai ecosystem, 2018.

[Yang *et al.*, 2019] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):12, 2019.

[Zhang and Zhu, 2020] Y. Zhang and Hao Zhu. Additively homomorphical encryption based deep neural network for asymmetrically collaborative machine learning. *ArXiv*, abs/2007.06849, 2020.

[Zhang *et al.*, 2018] Qiao Zhang, Cong Wang, Hongyi Wu, Chunsheng Xin, and Tran V. Phuong. Gelu-net: A globally encrypted, locally unencrypted deep neural network for privacy-preserved learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3933–3939. International Joint Conferences on Artificial Intelligence Organization, 7 2018.