Generic Adversarial Attack Framework Against Vertical Federated Learning

Yimin Liu[1,2], **Peng Jiang**[2,*]
[1]School of Computer Science and Technology, Beijing Institute of Technology, China
[2]School of Cyberspace Science and Technology, Beijing Institute of Technology, China
{3220220945, pengjiang}@bit.edu.cn

Abstract

Vertical federated learning (VFL) enables feature-level collaboration by incorporating scattered attributes from aligned samples, and allows each party to contribute its personalized input to joint training and inference. The injection of adversarial inputs can mislead the joint inference towards the attacker's will, forcing other benign parties to make negligible contributions and losing rewards regarding the importance of their contributions. However, most attacks require server model queries, subsets of complete test samples, or labeled auxiliary images from the training domain. These extra requirements are not practical for real-world VFL applications. In this paper, we propose PGAC, a novel and practical attack framework for crafting adversarial inputs to dominate joint inference, which does not rely on the above requirements. PGAC advances prior attacks by requiring only access to auxiliary images from non-training domains. PGAC learns generalized label-indicative embeddings and estimates class-transferable probabilities across domains to generate a proxy model that closely approximates the server model. PGAC then augments images by emphasizing salient regions with class activation maps, creating a diverse shadow input set that resembles influential test inputs. With proxy fidelity and input diversity, PGAC crafts transferable adversarial inputs. Evaluation on diverse model architectures confirms the effectiveness of PGAC.

1 Introduction

Federated learning (FL) has emerged as a promising technique for data silos, allowing multiple parties collaboratively train a model without disclosing their private data [McMahan *et al.*, 2017]. According to data distribution modes among various parties, FL can be categorized into horizontal federated learning (HFL) and vertical federated learning (VFL) [Hard *et al.*, 2018]. HFL involves clients with different data samples but a same feature space, while VFL applies to clients with same samples but complementary feature spaces [Wang *et al.*, 2023a]. As the availability of domain-specific image data increases and the need for collaboration at the feature level, VFL has gained prominence in many real-world image classification applications such as healthcare systems, e-commerce platforms, and IoT sensing, due to its advantage of incorporating scattered attributes from aligned samples [Poirot *et al.*, 2019; Mammen, 2021; Shi *et al.*, 2022]. Suppose that a service provider, referred to as the *active party*, owns image data and labels related to entities and wishes to train a server model for image classification. The active party may collaborate with other parties (i.e., *passive parties*) that possess complementary image features of the same entities. Instead of sharing raw image data and labels, the active party aggregates embeddings from passive parties' embedding models for joint training and subsequent inference [Bai *et al.*, 2023; Liu *et al.*, 2024].

While VFL ensures that each party's input contributes to business services, it is still vulnerable to adversarial inputs [Pang *et al.*, 2022; Qiu *et al.*, 2022], where subtle perturbations to a partial input feature from an adversarial party cause the server model to misclassify during joint inference. The existence of adversarial inputs reveals security risks in deploying VFL in real-world applications, enabling attackers to steer predictions toward a desired class and monopolize rewards, as only their input features contribute meaningfully to the joint inference [Sim *et al.*, 2020]. Despite the advancements in crafting adversarial inputs, they have to pre-set some restricted assumptions. For example, in [Pang *et al.*, 2022], it is required to query the server model and fetch a subset of complete test samples in advance. In practice, this resource-intensive query budget is costly and inevitably alerts the service provider. Additionally, acquiring input from other parties is difficult due to privacy concerns and transmission limitations. Another work [Qiu *et al.*, 2022] relies on labeled data sharing the same data distribution and class space as the training data. This is not consistent with real-world images which are usually with significant cross-domain disparities.

Motivated by above issues, we propose PGAC, a practical adversarial attack framework against VFL. PGAC enables a passive party to craft adversarial inputs that dominate joint inference by exploiting labeled auxiliary image data, even when it differs from the training data in both distribution and class space. PGAC consists of two modules: **P**roxy **G**eneration and **A**dversarial **C**rafting. The proxy generation module pro-

[*]Corresponding author: Peng Jiang

duces a high-fidelity proxy model that emulates the server model's behavior, enabling accurate gradient guidance to direct perturbations toward the direction of minimum classification loss for desired class. This is achieved by transferring label knowledge from a labeled source domain (i.e., auxiliary images) to an unlabeled target domain (i.e., training images). Through adversarial learning of transferable features, this module captures critical label-indicative characteristics across images and then aligns the cross-domain distributions within the shared class space while filtering out outlier classes using class transferable probabilities. The adversarial crafting module iteratively generates dominant perturbations that negate the influence of test input features from any other party during the server model's joint inference. This is achieved by constructing a diverse contribution-emphasized shadow input set, enforcing adversarial perturbation optimization that suppresses influential test inputs in the proxy's joint inference. Simultaneously, the high fidelity of the proxy model and the diversity of shadow inputs ensure crafted adversarial inputs can be transferred to dominate the server model's joint inference. Finally, we assess three potential countermeasures.

The main contributions of this paper are twofold.

- We propose PGAC, a novel adversarial attack framework against VFL that requires no queries to the server model, real test inputs from other parties, and labeled images from the training domain. PGAC introduces a high-fidelity proxy model and diverse influential shadow inputs for joint inference, crafting transferable adversarial inputs that effectively control the server model's predictions and diminish benign parties' contributions.
- We implement a PGAC prototype system based on the dual-module architecture, and conduct experiments to evaluate performance of PGAC. Extensive experiments across models confirm that PGAC dominates the VFL prediction, achieving 96.33% attack success rate, and outperforming state-of-the-art attacks by up to 31.83%. Results against three potential countermeasures also show that PGAC generally survives existing defenses.

2 Problem Statement

2.1 VFL Training and Inference

Consider an image classification task with m parties $\mathcal{P} = \{\mathcal{P}^1, \dots, \mathcal{P}^m\}$, each holding a training dataset $\mathbf{D}^i_{\text{train}} = (\mathcal{U}^i_{\text{train}}, \mathcal{F}^i)$ and a test dataset $\mathbf{D}^i_{\text{test}} = (\mathcal{U}^i_{\text{test}}, \mathcal{F}^i)$. Here $\mathcal{U}^i$ is the sample space, representing the set of images, and $\mathcal{F}^i$ is the feature space. Before training and inference, VFL determines the global sample space $\mathcal{U} = \{\mathcal{U}_{\text{train}}, \mathcal{U}_{\text{test}}\}$ as the intersection of local sample spaces [Chen *et al.*, 2017]: $\mathcal{U} = \bigcap_{i=1}^m \mathcal{U}^i$, and aligns features from different $\mathcal{F}^i$ for each sample.

In VFL, only one party holds the labels, referred to as the active party, while the others act as passive parties [Li *et al.*, 2023a; Fu *et al.*, 2022]. Each party $\mathcal{P}^i$ trains a local embedding model to map the d^i-dimensional feature vector $\mathbf{x}^i_{u_{\text{tra}}}$ of a training sample u_{tra}, from the feature space $\mathcal{F}^i$ of $\mathbf{D}^i_{\text{train}}$, into a latent embedding $\mathbf{h}^i_{u_{\text{tra}}}$ [Bai *et al.*, 2023]: $\mathcal{M}^i_e(\mathbf{x}^i_{u_{\text{tra}}}; \theta^i) : \mathbb{R}^{d^i} \to \mathbb{R}^{d^i_*}$, where $d^i = \text{he}^i \times \text{wi}^i \times \text{de}^i$, with he^i, wi^i, and de^i denoting the height, width, and depth of images owned by $\mathcal{P}^i$. Each party then uploads its embeddings $\mathbf{h}^i_{u_{\text{tra}}}$ to the server model $\mathcal{M}_s$, controlled by the active party. The embeddings are concatenated as $\mathbf{h}_{\text{cat}} = [\mathbf{h}^1_{u_{\text{tra}}}, \dots, \mathbf{h}^m_{u_{\text{tra}}}]$. The server model maps $\mathbf{h}_{\text{cat}}$ to $\mathbf{h}_{\text{top}}$: $\mathcal{M}_s(\mathbf{h}_{\text{cat}}; \theta_{\text{top}}) : \mathbb{R}^{d^1_* + \dots + d^m_*} \to \mathbb{R}^c$, where c is the number of classes. The output $\mathbf{h}_{\text{top}}$ is used to compute the loss, e.g., cross-entropy loss, to optimize the parameters of all models. The VFL training objective is formulated as:

$$\min_{\{\theta^i\}_{i=1}^m, \theta_{\text{top}}} \mathbb{E}_{u_{\text{tra}} \in \mathcal{U}_{\text{train}}} \left[\ell_{ce} \left(\mathbf{h}^1_{u_{\text{tra}}}, \dots, \mathbf{h}^m_{u_{\text{tra}}}, y_{u_{\text{tra}}}; \{\theta^i\}_{i=1}^m, \theta_{\text{top}} \right) \right],$$

where $\ell_{ce}(\cdot)$ is the loss function, and $y_{u_{\text{tra}}}$ is the label of u_{tra}. Each party then receives the gradient: $g^i_{u_{\text{tra}}} = \partial \ell_{ce} / \partial \mathbf{h}^i_{u_{\text{tra}}}$, to update its embedding model.

During inference, each party $\mathcal{P}^i$ computes the local embedding $\mathbf{H}^i_{u_{\text{test}}} = \mathcal{M}^i_e(\mathbf{X}^i_{u_{\text{test}}}; \theta^i)$ for the d^i-dimensional input feature vector $\mathbf{X}^i_{u_{\text{test}}}$ of test sample u_{test} in $\mathbf{D}^i_{\text{test}}$ [Pang *et al.*, 2022]. The server model concatenates all embeddings into $\mathbf{H}_{\text{cat}}$ and computes the prediction: $p = \mathcal{M}_s(\mathbf{H}_{\text{cat}}; \theta_{\text{top}})$, where p is a probability vector over c classes. The predicted label is determined as the class with the highest probability: $\hat{y}^s = \arg\max(p)$. For simplicity, we use $\mathbf{x}^i$, $\mathbf{X}^i$ denote $\mathbf{x}^i_{u_{\text{tra}} \in \mathcal{U}_{\text{train}}}$ and $\mathbf{X}^i_{u_{\text{test}} \in \mathcal{U}_{\text{test}}}$ in the following.

2.2 Adversarial Attacks in VFL

We assume that m parties have aligned their shared samples beforehand and each party's test inputs follow the distribution of their training inputs [Li *et al.*, 2023a; Qiu *et al.*, 2022; Pang *et al.*, 2022]. Among the remaining $m-1$ passive parties, one is an attacker $\mathcal{A}$, following the normal protocol but attempting to craft adversarial inputs that dominate the joint inference toward desired class l_{tar}.

Attacker's Capability & Knowledge. We assume that $\mathcal{A}$ can train a proxy version of server model $\mathcal{M}_s$ with auxiliary data and arbitrarily perturb its controlled test inputs $\mathbf{X}^{\mathcal{A}}$ in $\mathbf{D}^{\mathcal{A}}_{\text{test}}$.

$\mathcal{A}$ lacks access to $\mathcal{M}_s$ (including the architecture, parameters and query-based predictions), other parties' test inputs $\mathbf{X}^{\mathcal{B}}$, the embedding model $\mathcal{M}^{\mathcal{B}}_e$, and any labeled training data. $\mathcal{A}$ have its training dataset $\mathbf{D}^{\mathcal{A}}_{\text{train}}$ and test dataset $\mathbf{D}^{\mathcal{A}}_{\text{test}}$, optimized embedding model $\mathcal{M}^{\mathcal{A}}_e$ (including shared gradients $g^{\mathcal{A}}$) and access to a non-training domain labeled auxiliary dataset $\mathbf{D}_{\text{aux}}$. $\mathbf{D}_{\text{aux}}$ differs in data distribution from $\mathbf{D}^{\mathcal{A}}_{\text{train}}$ and encompasses $\mathbf{D}^{\mathcal{A}}_{\text{train}}$'s class space $\mathcal{C}_t$ within its broader class space $\mathcal{C}_s$. The exact overlap of class spaces between $\mathbf{D}_{\text{aux}}$ and $\mathbf{D}^{\mathcal{A}}_{\text{train}}$ (i.e., the shared class space $\mathcal{C} = \mathcal{C}_s \cap \mathcal{C}_t$ and the outlier class space $\bar{\mathcal{C}}_s = \mathcal{C}_s \setminus \mathcal{C}_t$) remains unknown to $\mathcal{A}$. This is realistic from large-scale data collection [Cao *et al.*, 2018]. For instance, when a training dataset includes specific disease categories, auxiliary datasets could come from larger medical institutions to encompass a wider range of diseases. To launch attacks, $\mathcal{A}$ is aware that class l_{tar} is included in $\mathcal{C}$.

Definition of Adversarial Attacks. Conventional attacks aim to misclassify a complete test input $\mathbf{X}^{\mathcal{A}}$ into a desired class l_{tar} [Zhou *et al.*, 2020]. The optimization objective is minimize loss function ℓ for l_{tar}:

$$\min_{\delta} \mathbb{E}_{\mathbf{X}^{\mathcal{A}} \in \mathbf{D}^{\mathcal{A}}_{\text{test}}} \left[\ell_{ce}(f(\mathbf{X}^{\mathcal{A}} + \delta), l_{tar}) \right], \quad \text{s.t. } \|\delta\|_p \leq \Lambda,$$

where f is the classification model, δ is the perturbation.

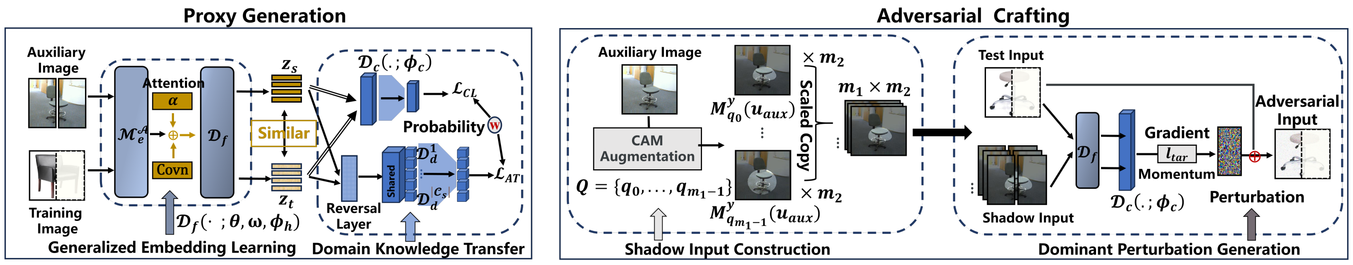

Figure 1: Overview of PGAC.

In VFL, each party holds only a partial feature space of test samples. Let $\mathbf{X}^{\mathcal{B}} \in \mathbf{D}^{\mathcal{B}}_{\text{test}}$ denote the inputs from all benign parties, i.e., $\mathbf{X}^{\mathcal{B}_1} \in \mathbf{D}^{\mathcal{B}_1}_{\text{test}}, \ldots, \mathbf{X}^{\mathcal{B}_{m-1}} \in \mathbf{D}^{\mathcal{B}_{m-1}}_{\text{test}}$. During inference, the server model $\mathcal{M}_s$ combines the attacker-controlled input $\mathbf{X}^{\mathcal{A}}$ with benign inputs $\mathbf{X}^{\mathcal{B}}$ to generate predictions. To dominate joint inference towards l_{tar} without being affected by changes in the benign inputs, adversarial inputs are crafted by minimizing the loss for l_{tar} while reducing the contributions of benign inputs, as measured by the saliency score. The optimization objective is formulated as:

$$\min_{\delta} \quad \mathbb{E}_{\mathbf{X}^{\mathcal{B}} \in \mathbf{D}^{\mathcal{B}}_{\text{test}}} \Big[\ell_{ce}\Big(\mathcal{M}_s(\mathcal{M}^{\mathcal{A}}_e(\mathbf{X}^{\mathcal{A}} + \delta), \mathcal{M}^{\mathcal{B}_1}_e(\mathbf{X}^{\mathcal{B}_1}), \ldots, \mathcal{M}^{\mathcal{B}_{m-1}}_e(\mathbf{X}^{\mathcal{B}_{m-1}})), l_{tar} \Big) + \textbf{\textit{Saliency}}(\mathbf{X}^{\mathcal{B}}) \Big], \text{s.t. } \|\delta\|_p \leq \Lambda.$$

Definition 1 (Saliency Score). *Given a server model $\mathcal{M}_s$'s prediction: output $= \mathcal{M}_s(\mathcal{M}^{\mathcal{A}}_e(\mathbf{X}^{\mathcal{A}} + \delta), \mathcal{M}^{B}_e(\mathbf{X}^{B})))$, the saliency score for $\mathbf{X}^{\mathcal{B}_i}$ is defined as the l_1-norm of the derivative of the output variance with respect to $\mathbf{X}^{\mathcal{B}_i}$:*

$$\textbf{\textit{Saliency}}(\mathbf{X}^{\mathcal{B}_i}) = \left\| \frac{\partial}{\partial \mathbf{X}^{\mathcal{B}_i}} \operatorname{Var}(\textit{output}) \right\|_1 .$$

3 PGAC: Detailed Construction

As shown in Figure 1, PGAC comprises two modules. The proxy generation module creates a high-fidelity proxy model, allowing the use of accurate gradients to produce perturbations toward the direction of minimum loss to desired class. The adversarial crafting module iteratively generates dominant perturbations against diverse, contribution-emphasized shadow inputs to minimize their influence on the proxy joint inference, thereby crafting adversarial inputs.

3.1 Proxy Generation

Given the source (i.e., auxiliary) dataset $\mathbf{D}_{\text{aux}}$ and the target (i.e., training) dataset $\mathbf{D}^{\mathcal{A}}_{\text{train}}$, inspired by domain adversarial networks [Ganin *et al.*, 2016], we extract transferable features that mitigate domain shift. Specifically, the attacker's embedding model $\mathcal{M}^{\mathcal{A}}_e$ is employed as a feature encoder $\mathcal{D}_f(.;\theta)$ to learn domain-invariant features, while a domain discriminator $\mathcal{D}_d(.;\phi_d)$ learns to distinguish between domains. Simultaneously, a proxy model $\mathcal{D}_c(.;\phi_c)$ is trained to categorize data.

Generalized Embedding Learning. Considering that relying solely on encoder parameters for domain-invariant embedding learning may inadequately shift distributions, we incorporate an attention layer into the encoder $\mathcal{D}_f$. This mechanism enforces $\mathcal{D}_f$ to characterize label-indicative features from raw images and learn the generalized critical embedding to promote better alignment across domains.

Formally, the raw embedding produced by $\mathcal{D}_f$ is denoted as $\mathbf{h} = \{\mathbf{h}_s, \mathbf{h}_t\} \in \mathbb{R}^{d_*}$, where d_* is the dimension, and $\mathbf{h}_s$, $\mathbf{h}_t$ are the embeddings of source feature $\mathbf{x}_{\text{aux}}$ of image u_{aux} in $\mathbf{D}_{\text{aux}}$ and target feature $\mathbf{x}^{\mathcal{A}}$ in $\mathbf{D}^{\mathcal{A}}_{\text{train}}$, respectively. (Here, each u_{aux} is pre-split to ensure alignment between the feature dimensions of $\mathbf{x}_{\text{aux}}$ and $\mathbf{x}^{\mathcal{A}}$.) To emphasize category-specific information, an attention score is computed to assess the importance of $\mathbf{h}$ relative to each class:

$$\alpha^c = \frac{\exp(\mathbf{h}^T \omega^c / \tau)}{\sum_{c'} \exp(\mathbf{h}^T \omega^{c'} / \tau)},$$

where $\omega^c \in \mathbb{R}^{d_*}$ is a learnable query vector for class $c \in |\mathcal{C}_s|$, and τ is a temperature coefficient. Using the attention score α^c, the embedding under class c is calculated as $\hat{\mathbf{h}}^c = \alpha^c \mathbf{h}$. The generalized critical embedding $\mathbf{z}$ is then obtained by applying a convolution operation:

$$\mathbf{z} = \text{Conv}([\hat{\mathbf{h}}^1, \ldots, \hat{\mathbf{h}}^{|\mathcal{C}_s|}]; \phi_h),$$

where $\hat{\mathbf{h}}^c \in \mathbb{R}^{d_*}$ is the latent vector for class c, and $\phi_h \in \mathbb{R}^{|\mathcal{C}_s| \times 1}$ represents the convolution layer parameters. The learned critical embeddings $\mathbf{z} = \{\mathbf{z}_s, \mathbf{z}_t\} \in \mathbb{R}^{d_*}$, with their respective distributions $\mathcal{Z}_s$ and $\mathcal{Z}_t$ corresponding to $\mathbf{h}_s$ and $\mathbf{h}_t$, are subsequently utilized for adapting the proxy model.

Domain Knowledge Transfer. Since we can not know which part of the source class space $\mathcal{C}_s$ is shared with the target class space $\mathcal{C}_t$, to avoid potential negative transfer caused by using a single domain discriminator to match the entire source and target distribution, we employ a multi-task discriminator with shared bottom layers and $|\mathcal{C}_s|$ heads. Each head $\mathcal{D}^{(k)}_d$ (for $k = 1, \ldots, |\mathcal{C}_s|$) enables conditional alignment between the source and target embeddings associated with the k-th source class. With the multi-task discriminator, we assign each instance $\mathbf{z}$ to the correct alignment task by computing the predicted probability distribution over the source class space $\hat{p} = \mathcal{D}_c(\mathbf{z}, \mathbf{z}; \phi_c)$, where $\mathcal{D}_c$ is a proxy model, $\mathbf{z}$ is stacked as a double input to simulate a two-party VFL setting, mapping $\mathbb{R}^{2 \times d_*} \to \mathbb{R}^{|\mathcal{C}_s|}$. $\hat{p}$ can characterize the probability of assigning $\mathbf{z}$ to each of the $|\mathcal{C}_s|$ source classes [Saito *et al.*, 2017], thus we embed $\hat{p}$ as an instance weight in the domain discriminator loss for all $|\mathcal{C}_s|$ heads.

To mitigate the impact of outlier source classes $k \in \bar{\mathcal{C}}_s$, where the corresponding discriminator $\mathcal{D}^{(k)}_d$ processes only source data and provides irrelevant signals, we calculate the transferable probability for each source class to appropriately down-weight the domain discriminator responsible for these outlier classes. The class transferable probability is derived from the predictions of the target data $\mathbf{z}_t$, considering that target data are unlikely to belong to these outlier classes. This is formulated as $\mathbf{w} = \mathbb{E}_{\mathbf{z}_t \sim \mathcal{Z}_t}[\mathcal{D}_c(\mathbf{z}_t, \mathbf{z}_t; \phi_c)]$. Similar to CDAN [Long *et al.*, 2018], we further refine the discriminator focus on easily transferable data with certain predictions by applying a secondary weighting using an entropy-aware weight $w_e(\mathbf{z}) = 1 + e^{-J(\mathcal{D}_c(\mathbf{z},\mathbf{z};\phi_c))}$, where J denotes the entropy function. Final discriminator loss is formulated as:

$$\mathcal{L}_{AT} = \sum_{k=1}^{|\mathcal{C}_s|} \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}_s \cup \mathcal{Z}_t}\Big[\mathbf{w}_k \cdot w_e(\mathbf{z})\hat{p}^k \ell_{\text{ce}}(\mathcal{D}_d^{(k)}(\mathbf{z}; \phi_d), d)\Big],$$

where $\hat{p}^k$ is the k-th entry of prediction for $\mathbf{z}$, and $\mathbf{w}_k$ is the k-th entry of $\mathbf{w}$, indicating the probability of source class k falling into shared class space $\mathcal{C}$. The domain discriminator $\mathcal{D}_d$ determines whether $\mathbf{z}$ originates from the source or target domain. d is the domain label (0 for target and 1 for source).

$\mathcal{L}_{AT}$ relies on the prediction of proxy model $\mathcal{D}_c$ on both source and target domain. To reduce the probability of predicting a label in the outlier source classes $k \in \bar{\mathcal{C}}_s$, we embed the class transferable probability $\mathbf{w}$ into the supervised loss (with label y) of $\mathcal{D}_c$ to focus on the source data from the shared class space $\mathcal{C}$. To reduce the uncertainty of target predictions, we introduce a self-training loss to further refine $\mathcal{D}_c$ [Xie *et al.*, 2020]. We iteratively assign pseudo labels $\hat{y}^c_{\text{self}} = \arg\max(\mathcal{D}_c(\mathbf{z}_t, \mathbf{z}_t; \phi_c))$ to unlabeled target data $\mathbf{z}_t$, and embed $\mathbf{w}$ into the self-training loss of $\mathcal{D}_c$ to confine predictions also remain within $\mathcal{C}$. We then introduce gradient similar loss ℓ_2 to emphasize $\mathcal{D}_c$'s fidelity to the server model by minimizing discrepancy between the proxy gradient g_t and the original shared $g_t^{\mathcal{A}}$ for the raw input $\mathbf{h}_t$, formulated as: $\ell_2(g_t^{\mathcal{A}}, g_t) = \left\|g_t^{\mathcal{A}} - \partial\ell_{\text{ce}}(\mathcal{D}_c(\mathbf{z}_t, \mathbf{z}_t; \phi_c), \hat{y}^c_{\text{self}})/\partial\mathbf{h}_t\right\|_2$. Thus, the proxy model loss $\mathcal{L}_{CL}$ is formulated as:

$$\begin{aligned}\mathcal{L}_{CL} &= \mathbb{E}_{\mathbf{z}_s \sim \mathcal{Z}_s}\left[\mathbf{w}_y \cdot \ell_{\text{ce}}\left(\mathcal{D}_c(\mathbf{z}_s, \mathbf{z}_s; \phi_c), y\right)\right] \\ &+ \lambda_1 \mathbb{E}_{\mathbf{z}_t \sim \mathcal{Z}_t}\left[\mathbf{w}_{\hat{y}^c_{\text{self}}} \cdot \ell_{\text{ce}}(\mathcal{D}_c(\mathbf{z}_t, \mathbf{z}_t; \phi_c), \hat{y}^c_{\text{self}}) + \ell_2(g_t^{\mathcal{A}}, g_t)\right].\end{aligned}$$

Optimization. Integrating all losses, the objective of proxy generation can be formulated as:

$$\min_{\theta,\omega,\phi_h,\phi_c}\left(\mathcal{L}_{CL} - \lambda_2 \max_{\phi_d}\{\mathcal{L}_{AT}\}\right).$$

Following [Ganin *et al.*, 2016], we frame the problem as a min-max optimization game to find the optimal parameters.

3.2 Adversarial Crafting

Given each complete image u_{aux} in $\mathbf{D}_{\text{aux}}$, we simulate a latent party $\mathcal{B}'$ holding influential test inputs during joint inference. Specifically, regions in u_{aux} that contribute to classification are emphasized, then duplicated and scaled to construct a diverse shadow input set $\mathcal{S}$ for $\mathcal{B}'$. Building on the proxy model $\mathcal{D}_c$ and $\mathcal{S}$, adversarial perturbations are iteratively added to $\mathbf{X}^{\mathcal{A}}$, optimized to diminish the input contributions of $\mathcal{B}'$.

Shadow Input Construction. We apply gradient-weighted class activation mapping (Grad-CAM) [Selvaraju *et al.*, 2017] to compute CAM heatmap matrix $L_f^y(u_{\text{aux}})$, identifying each pixel's contribution to label y:

$$L_f^y(u_{\text{aux}}) = H\left(\text{ReLU}\left(\sum_n \beta_n^y F_n\right)\right),$$

where F_n is the n-th channel of the feature map from the final convolutional layer of the embedding model $\mathcal{M}_e^{\mathcal{A}}$, $H(\cdot)$ performs bilinear interpolation to align CAM dimensions with the input, and $\beta_n^y = \frac{1}{i \cdot j}\sum_i \sum_j \frac{\partial y}{\partial F_n^{ij}}$ represents the contribution weights for each feature map channel.

To emphasize salient regions, pixels below the q-th percentile of $L_f^y(.)$ can be suppressed as:

$$C_q^y(.) = L_f^y(.) \cdot \max\left(\text{Sign}\left(Q_q(L_f^y(.)) - L_f^y(.)\right), 0\right),$$

where Q_q is a matrix of the same dimensions as input, with each element set to the q-th percentile value of all pixels in $L_f^y(.)$. The masked heatmap $C_q^y(u_{\text{aux}})$ is then fused with u_{aux} to produce contribution-emphasized data $M_q^c(u_{\text{aux}})$:

$$M_q^y(u_{\text{aux}}) = (1 - \mu) \cdot u_{\text{aux}} + \mu \cdot C_q^y(u_{\text{aux}}) + \xi,$$

where $\mu \in [0, 1]$ is the mixing ratio, and ξ is random noise added for diversity. Since $C_q^y(u_{\text{aux}})$ is generated based on $\mathbf{x}$, the original label is directly used for $M_q^y(u_{\text{aux}})$.

To enhance the transferability of crafted adversarial input, we duplicate and scale $M_q^y(u_{\text{aux}})$ at varying levels to construct the shadow set $\mathcal{S}$. A percentile set Q is formulated as:

$$Q = \{q \mid q \bmod \epsilon = 0,\ q \in \mathbb{N},\ q < 100\},$$

where q is the percentile threshold for masking, and ϵ determines granularity. Let m_1 and m_2 denote the number of percentile thresholds and scaling factors, respectively. The shadow input set $\mathcal{S} = \bigcup \mathcal{S}_{\text{aux}}$ is constructed as:

$$\mathcal{S}_{\text{aux}} = \{M_{q_i}^y(u_{\text{aux}})/2^j \mid i \in [0, m_1 - 1], q_i \in Q, j \in [0, m_2 - 1]\}.$$

Dominant Perturbation Generation. With $\mathcal{D}_c$ and $\mathcal{S}$, adversarial perturbations are generated using a gradient-based method [Madry *et al.*, 2017] to dominate joint inference. The optimization is constrained to make $\mathcal{A}$'s test input $\mathbf{X}^{\mathcal{A}}$ disregard the influence of other inputs, as quantified with their saliency scores (**Definition 1**). As $\mathcal{A}$ lacks access to real parties' embedding models, the saliency score for $\mathcal{B}'$ are estimated via FDM [Ciesielski and Leszczynski, 2006], which computes the derivative of output variance with respect to $\mathcal{S}$.

Algorithm 1 outlines the adversarial input crafting process. A penalty term ensures subtle perturbations constrained to predefined ranges, while momentum [Dong *et al.*, 2018] accelerates updates toward optimal directions. The optimization problem is solved with projected gradient descent (PGD) [Goodfellow *et al.*, 2015]. Leveraging the proxy fidelity and input diversity, crafted adversarial input effectively transfer to disrupt the server model $\mathcal{M}_s$'s joint inference.

4 Evaluation

4.1 Setting

Datasets & VFL Configuration. We evaluate PGAC on three cross-domain image classification datasets following [Gu *et al.*, 2021]. *Office-31* contains 4,652 images of 31 categories, collected from three domains: Amazon (A), DSLR (D), and Webcam (W) [Saenko *et al.*, 2010]. We select images from the 10 categories shared by Office-31 and Caltech-256 [Griffin *et al.*, 2007] to build the VFL dataset, following [Cao *et al.*, 2018]. *ImageNet-Caltech* is created with

Algorithm 1 Adversarial Input Crafting

```
1:  function SALIENCY_EST($\mathbf{X}^{\mathcal{A}}, \mathbf{X}^{\mathcal{B}'}, \mathcal{D}_f, \mathcal{D}_c$)
2:      $output \leftarrow \mathcal{D}_c(\mathcal{D}_f(\mathbf{X}^{\mathcal{A}}), \mathcal{D}_f(\mathbf{X}^{\mathcal{B}'}))$
3:      $output_{\delta_B} \leftarrow \mathcal{D}_c(\mathcal{D}_f(\mathbf{X}^{\mathcal{A}}), \mathcal{D}_f(\mathbf{X}^{\mathcal{B}'} + \delta_B))$
4:      $saliency \leftarrow \frac{\mathrm{Var}(output_{\delta_B}) - \mathrm{Var}(output)}{\delta_B}$
                    ▷ Var(·): variance; $\delta_B$: a small perturbation.
5:      return $saliency$
6:  end function
7:  function AI_CRAFTING($\mathbf{X}^{\mathcal{A}}, \mathcal{D}_f, \mathcal{D}_c, l_{tar}, \mathcal{S}$)
                    ▷ $\alpha, \beta, \gamma, \sigma$: weights; $\Lambda$: constraint; $V$: mutation.
8:      $V \leftarrow 0, \delta_0 \leftarrow 0, t \leftarrow 1$
9:      while $t \leq T$ do
10:         for each $M_{q_i}^{y}(u_{\text{aux}})/2^j \in \mathcal{S}$ do
11:             $\delta_t \leftarrow \arg\min_{\delta_t} \alpha \cdot$ SALIENCY_EST($\mathbf{X}^{\mathcal{A}} + V + \delta_t, M_{q_i}^{y}(u_{\text{aux}})/2^j, \mathcal{D}_f, \mathcal{D}_c$)
                 $+ \beta \cdot \ell_{ce}(\mathcal{D}_c(\mathcal{D}_f(\mathbf{X}^{\mathcal{A}} + V + \delta_t), \mathcal{D}_f(M_{q_i}^{y}(u_{\text{aux}})/2^j)), l_{tar})$
                 $+ \gamma \|\delta_t\|_2$,  s.t. $\|V + \delta_t\| \leq \Lambda$
12:             $\delta_t \leftarrow \sigma \cdot \delta_{t-1} + \delta_t, V \leftarrow V + \delta_t, t \leftarrow t + 1$
13:         end for
14:     end while
15:     return $\mathbf{X}^{\mathcal{A}} + V$
16: end function
```

ImageNet (I) [Russakovsky *et al.*, 2015] (1,000 classes) and the Caltech-256 (C) (256 classes). We utilize the 84 shared classes to build the VFL dataset. *DomainNet* composed of six domains with 345 classes [Peng *et al.*, 2019]. We adopt four domains (Clipart (C), Painting (P), Real (R), and Sketch (S)) with 126 classes, and use the first 40 classes in alphabetical order to build the VFL dataset [Saito *et al.*, 2019].

Our default experiments adopt the common setting with two parties, each owning half of the sample's feature vector within the VFL dataset [Qiu *et al.*, 2022; Pang *et al.*, 2022]. We sample the raw image dataset from domains with broader class spaces as the attacker's auxiliary data, selecting a default of 100 samples per class. Table 1 summarizes the VFL dataset and auxiliary data for each image dataset.

Models. We adopt widely used architectures as the *embedding models*, including VGG-16, VGG-19 [Simonyan, 2014], ResNet-50, ResNet-152 [He *et al.*, 2016], DenseNet-121, and DenseNet-201 [Huang *et al.*, 2017], all excluding their final fully connected layers. By default, VGG-16 is used for each party. The *server model* is implemented as a two-layer MLP, while the *proxy model* $\mathcal{D}_c$ consists of three hidden layers with 32 neurons each. The *domain discriminator* $\mathcal{D}_d$ is a single-layer MLP with 16 neurons. The proxy generation process spans 200 epochs, with the loss weight λ_1 set to 1, and λ_2 linearly increasing each epoch as $\lambda_2 = \frac{t}{200}$, where t is the current epoch. For shadow input construction, the percentile parameter ϵ, number of scaling factors m_2, and mix ratio μ are set to 10, 5, and 0.6, respectively. We configure the crafting process with a maximum perturbation budget of $\Lambda = 15$ and $T = 20$ iterations, inspired by [Xie *et al.*, 2018].

Metrics. PGAC is evaluated with the attack success rate (ASR) and fidelity. ASR measures the effectiveness of dominating server's joint inference, calculated as: $\frac{1}{n}\sum_{i=1}^{n} I\left(\mathcal{M}_s(\mathcal{D}_f(\mathbf{X}^{\mathcal{A}} + V), \mathcal{M}_e^{\mathcal{B}}(\mathbf{X}^{\mathcal{B}}) = l_{\text{tar}}\right)$, where n is the number of test samples, $\mathbf{X}^{\mathcal{B}}$ denote the inputs from all benign parties, and $I(\cdot)$ is the indicator function. Fidelity quantifies the agreement between predictions from the proxy model $\mathcal{D}_c$ and server model $\mathcal{M}_s$, calculated as: $\frac{1}{n}\sum_{i=1}^{n} I\left(\hat{y}^c = \hat{y}^s\right)$, where n is the number of test samples, $\hat{y}^c$ and $\hat{y}^s$ are the prediction of $\mathcal{D}_c$ and $\mathcal{M}_s$, respectively.

Baselines. We use real labeled training data to train a proxy model and use test data from real parties for crafting inputs, establishing the Real Data benchmark. We then compare PGAC against state-of-the-art attacks. For [Pang *et al.*, 2022], we replace the real test data with our cross-domain auxiliary data and train a proxy model on it without querying the server model. For [Qiu *et al.*, 2022], we replace the labeled training data with our auxiliary data for proxy training while retaining their poisoning process. Additionally, we benchmark PGAC against transfer-based methods using pre-trained models, including MIFGSM, SINIFGSM and DIFGSM [Dong *et al.*, 2018; Lin *et al.*, 2020; Xie *et al.*, 2018].

All experiments are conducted on a workstation equipped with an Intel Core i7-10700K processor and running Ubuntu 20.04.1 LTS. Each experiment is repeated five times, and we report the average ASR and fidelity.

4.2 Overall Performance

Table 2 reports ASR and fidelity, results show that the Real Data benchmark achieves the highest performance, while PGAC approaches this benchmark with deviations of at most 4.26% in ASR and 4.98% in fidelity. PGAC consistently outperforms the state-of-art attacks, exhibiting improvements of at least 25.78% in ASR and 43.98% in fidelity. This superiority stems from addressing cross-domain discrepancies and mining salient regions for classification. PGAC also surpasses the transfer-based methods with at least 29.11% higher ASR, indicating the importance of training proxy models that closely approximate server models for attacks.

4.3 Parameter Analysis

Impact of Embedding & Server Models. We evaluate PGAC's performance by varying the embedding model, server model for the active party and attacker, respectively. As shown in Tables 3 and 4, PGAC consistently demonstrates effectiveness across different model architectures.

Impact of Auxiliary Data Volume. Figure 2(a) shows that increasing the auxiliary data improves both ASR and fidelity. On the Office-31, with just 20 labeled samples per class, PGAC achieves an ASR of 85.31% and fidelity of 82.04%. However, the improvement gradually stabilizes when the sample number exceeds 100 per class. Notably, with only 120 samples per class, far below the VFL dataset size, PGAC achieves an ASR of 97.82% and fidelity of 94.71%.

Impact of Party Numbers. We expand typical multi-party VFL settings from fewer than 16 benign passive parties to 32 [Qiu *et al.*, 2022; Pang *et al.*, 2022]. We fix the active party's feature ratio of 50%, and the remaining features are vertically partitioned between the attacker and added passive parties. Results and corresponding data setup are detailed in Table 5. PGAC consistently achieves ASR and fidelity above 85.18% and 82.12% on Office-31, even with up to 32 parties.

4.4 Adaptation Analysis and Ablation Study

We evaluate PGAC's adaptation in proxy generation and component utility with experiments on the Office-31 dataset.

Dataset	**Feature Partition**		**Auxiliary Dataset → VFL Dataset**	**Auxiliary Class Space → VFL Class Space**
	Active Party	Attacker		
Office-31	50%	50%	A → W, D → W, W → D, A → D, D → A, W → A	Full 31 classes → 10 classes shared with Caltech-256
ImageNet-Caltech	50%	50%	I → C, C → I	Full 256 classes → 84 classes shared by ImageNet and Caltech-256
DomainNet	50%	50%	C→P, C→R, C→S, P→C, P→R, P→S, R→C, R→P, R→S, S→C, S→P, S→R	Full 345 classes → First 40 classes in alphabetical order

Table 1: VFL configuration. Feature partition refers to the proportion of vertical dimensions of an image assigned to each party.

Dataset	**Real Data**		**PGAC**		**[Pang *et al.*, 2022]**		**[Qiu *et al.*, 2022]**		**MIFGSM**	**SINIFGSM**	**DIFGSM**
	ASR	Fidelity	ASR	Fidelity	ASR	Fidelity	ASR	Fidelity	ASR	ASR	ASR
Office-31	99.85$_{\pm0.73}$	95.96$_{\pm0.51}$	95.60$_{\pm1.13}$	91.84$_{\pm0.77}$	63.77$_{\pm1.67}$	43.28 $_{\pm1.37}$	65.60$_{\pm1.30}$	44.13$_{\pm2.00}$	51.60$_{\pm2.33}$	56.60$_{\pm1.94}$	60.07$_{\pm1.05}$
ImageNet-Caltech	97.73$_{\pm0.55}$	92.39$_{\pm0.64}$	93.83$_{\pm0.77}$	87.81$_{\pm0.80}$	62.94$_{\pm1.04}$	42.28 $_{\pm1.50}$	68.05$_{\pm1.30}$	43.83$_{\pm1.32}$	50.11$_{\pm2.51}$	54.15$_{\pm1.19}$	61.15$_{\pm1.20}$
DomainNet	92.85$_{\pm1.08}$	88.17$_{\pm0.51}$	89.44$_{\pm1.50}$	83.78$_{\pm0.53}$	62.72$_{\pm1.70}$	39.10 $_{\pm1.28}$	61.89$_{\pm0.78}$	38.04$_{\pm1.10}$	52.11$_{\pm2.70}$	55.60$_{\pm1.18}$	60.33$_{\pm1.14}$

Table 2: PGAC's performance compared with baselines.

$\mathcal{M}_e^{\mathcal{B}}$ / $\mathcal{M}_e^{\mathcal{A}}$		**VGG-16**		**VGG-19**		**Res-50**		**Res-152**		**Dense-121**		**Dense-201**	
	Dataset	**ASR**	**Fidelity**	**ASR**	**Fidelity**	**ASR**	**Fidelity**	**ASR**	**Fidelity**	**ASR**	**Fidelity**	**ASR**	**Fidelity**
VGG-19	Office-31	92.95$_{\pm1.03}$	89.33$_{\pm1.23}$	96.33$_{\pm1.10}$	92.42$_{\pm1.45}$	90.30$_{\pm1.15}$	86.21$_{\pm1.41}$	91.71$_{\pm1.05}$	87.70$_{\pm1.27}$	93.56$_{\pm1.34}$	90.05$_{\pm1.17}$	92.00$_{\pm1.35}$	87.98$_{\pm1.02}$
	ImageNet-Caltech	90.75$_{\pm1.27}$	85.13$_{\pm1.39}$	94.67$_{\pm1.23}$	88.43$_{\pm1.38}$	88.10$_{\pm1.22}$	82.91$_{\pm1.57}$	89.36$_{\pm1.19}$	83.43$_{\pm1.35}$	91.36$_{\pm1.27}$	86.65$_{\pm1.18}$	90.80$_{\pm1.41}$	83.78$_{\pm1.08}$
	DomainNet	86.55$_{\pm1.43}$	80.01$_{\pm1.59}$	90.47$_{\pm1.38}$	82.02$_{\pm1.27}$	83.88$_{\pm1.32}$	77.10$_{\pm1.63}$	85.16$_{\pm1.28}$	77.84$_{\pm1.41}$	87.16$_{\pm1.34}$	80.10$_{\pm1.24}$	86.60$_{\pm1.50}$	77.99$_{\pm1.07}$
Res-50	Office-31	90.96$_{\pm1.12}$	86.33$_{\pm1.57}$	91.70$_{\pm1.22}$	87.61$_{\pm1.05}$	96.00$_{\pm1.10}$	92.15$_{\pm1.25}$	92.11$_{\pm0.85}$	87.70$_{\pm1.30}$	92.49$_{\pm1.74}$	88.25$_{\pm1.57}$	90.54$_{\pm1.55}$	86.05$_{\pm1.35}$
	ImageNet-Caltech	88.75$_{\pm1.28}$	82.54$_{\pm1.82}$	89.57$_{\pm1.36}$	83.44$_{\pm1.20}$	94.12$_{\pm1.32}$	88.11$_{\pm1.49}$	90.36$_{\pm0.93}$	83.45$_{\pm1.26}$	89.87$_{\pm1.63}$	83.09$_{\pm1.62}$	88.13$_{\pm1.58}$	82.83$_{\pm1.37}$
	DomainNet	84.92$_{\pm1.21}$	78.12$_{\pm1.96}$	85.62$_{\pm1.56}$	79.77$_{\pm1.43}$	90.89$_{\pm1.48}$	83.74$_{\pm1.61}$	86.51$_{\pm1.02}$	80.39$_{\pm1.34}$	86.72$_{\pm1.85}$	79.02$_{\pm1.74}$	84.75$_{\pm1.64}$	78.81$_{\pm1.44}$
Dense-121	Office-31	91.85$_{\pm1.06}$	88.00$_{\pm1.19}$	90.45$_{\pm1.08}$	86.56$_{\pm1.40}$	90.93$_{\pm1.18}$	87.05$_{\pm1.38}$	90.07$_{\pm1.02}$	86.45$_{\pm1.22}$	96.30$_{\pm1.12}$	92.00$_{\pm1.20}$	94.67$_{\pm1.12}$	90.02$_{\pm1.29}$
	ImageNet-Caltech	89.65$_{\pm1.22}$	82.88$_{\pm1.41}$	88.50$_{\pm1.21}$	82.17$_{\pm1.58}$	88.91$_{\pm1.33}$	82.00$_{\pm1.50}$	87.80$_{\pm1.14}$	82.23$_{\pm1.35}$	94.37$_{\pm1.14}$	88.58$_{\pm1.26}$	92.31$_{\pm1.15}$	85.75$_{\pm1.45}$
	DomainNet	85.71$_{\pm1.34}$	79.13$_{\pm1.56}$	84.14$_{\pm1.10}$	78.03$_{\pm1.62}$	84.75$_{\pm1.22}$	78.31$_{\pm1.61}$	83.87$_{\pm1.01}$	78.51$_{\pm1.49}$	90.58$_{\pm1.15}$	82.58$_{\pm1.35}$	88.16$_{\pm1.23}$	80.81$_{\pm1.53}$

Table 3: PGAC performance with varying embedding models. $\mathcal{M}_e^{\mathcal{B}}$ and $\mathcal{M}_e^{\mathcal{A}}$ are models used by the active party and attacker, respectively.

$\mathcal{M}_s$ / $\mathcal{D}_c$		**2-layer MLP**		**3-layer MLP**		**4-layer MLP**		**5-layer MLP**		**6-layer MLP**		**7-layer MLP**	
	Dataset	**ASR**	**Fidelity**	**ASR**	**Fidelity**	**ASR**	**Fidelity**	**ASR**	**Fidelity**	**ASR**	**Fidelity**	**ASR**	**Fidelity**
2-layer MLP	Office-31	96.17$_{\pm1.91}$	93.14$_{\pm1.89}$	94.50$_{\pm2.07}$	90.44$_{\pm2.08}$	93.91$_{\pm2.12}$	90.33$_{\pm2.34}$	95.42$_{\pm2.11}$	91.13$_{\pm2.33}$	93.47$_{\pm2.10}$	88.20$_{\pm2.19}$	96.24$_{\pm2.07}$	91.18$_{\pm2.25}$
	ImageNet-Caltech	95.75$_{\pm1.79}$	89.40$_{\pm2.10}$	92.00$_{\pm2.13}$	86.44$_{\pm1.78}$	91.22$_{\pm2.09}$	86.12$_{\pm2.28}$	92.57$_{\pm2.03}$	87.55$_{\pm2.21}$	90.62$_{\pm2.29}$	84.36$_{\pm1.83}$	93.16$_{\pm2.08}$	87.33$_{\pm2.22}$
	DomainNet	91.58$_{\pm2.05}$	82.73$_{\pm2.16}$	88.12$_{\pm1.87}$	80.44$_{\pm2.11}$	87.47$_{\pm1.88}$	79.81$_{\pm2.05}$	89.42$_{\pm2.01}$	80.81$_{\pm1.96}$	87.22$_{\pm2.16}$	79.93$_{\pm2.10}$	89.86$_{\pm1.74}$	82.83$_{\pm1.78}$
4-layer MLP	Office-31	94.03$_{\pm1.85}$	91.46$_{\pm1.98}$	96.28$_{\pm1.77}$	91.08$_{\pm1.92}$	96.01$_{\pm1.95}$	90.02$_{\pm2.12}$	93.11$_{\pm1.95}$	93.05$_{\pm1.87}$	93.68$_{\pm1.50}$	88.10$_{\pm1.45}$	94.11$_{\pm1.80}$	89.86$_{\pm1.92}$
	ImageNet-Caltech	93.70$_{\pm1.53}$	87.44$_{\pm1.71}$	94.12$_{\pm1.55}$	87.18$_{\pm1.65}$	95.68$_{\pm1.67}$	86.37$_{\pm1.89}$	91.11$_{\pm1.58}$	89.35$_{\pm1.53}$	91.56$_{\pm1.77}$	85.68$_{\pm1.85}$	92.45$_{\pm1.78}$	86.52$_{\pm1.26}$
	DomainNet	89.32$_{\pm1.64}$	81.04$_{\pm1.79}$	90.63$_{\pm1.58}$	81.38$_{\pm1.70}$	91.10$_{\pm1.72}$	78.97$_{\pm1.57}$	87.46$_{\pm1.62}$	82.57$_{\pm1.87}$	87.56$_{\pm1.74}$	79.57$_{\pm1.76}$	88.03$_{\pm1.70}$	80.31$_{\pm1.88}$
6-layer MLP	Office-31	95.30$_{\pm1.89}$	90.28$_{\pm2.01}$	93.45$_{\pm1.83}$	89.72$_{\pm1.98}$	92.11$_{\pm1.92}$	89.12$_{\pm2.07}$	94.73$_{\pm1.88}$	90.04$_{\pm1.99}$	95.78$_{\pm2.05}$	92.05$_{\pm1.66}$	93.56$_{\pm1.78}$	87.32$_{\pm1.94}$
	ImageNet-Caltech	93.38$_{\pm1.95}$	86.81$_{\pm2.07}$	91.01$_{\pm1.76}$	85.72$_{\pm1.93}$	90.11$_{\pm1.87}$	85.01$_{\pm1.84}$	92.32$_{\pm1.79}$	86.44$_{\pm1.91}$	93.55$_{\pm1.97}$	88.50$_{\pm1.89}$	91.60$_{\pm1.51}$	83.44$_{\pm1.98}$
	DomainNet	89.12$_{\pm1.92}$	81.08$_{\pm2.11}$	87.25$_{\pm1.68}$	80.02$_{\pm1.89}$	87.91$_{\pm2.05}$	79.56$_{\pm2.14}$	88.12$_{\pm1.76}$	81.52$_{\pm1.60}$	90.14$_{\pm1.88}$	81.87$_{\pm1.92}$	87.17$_{\pm1.40}$	78.63$_{\pm1.46}$

Table 4: PGAC performance with varying server models. $\mathcal{M}_s$ and $\mathcal{D}_c$ are models used by the active party and attacker, respectively.

Performance					Feature Partition		
Dataset	**# pa.**	**ASR**	**Fidelity**	**ori.acc.**	Active Party	Passive Party$_{1,\ldots,\text{\# pa.}}$	Attacker
Office-31	2	92.31$_{\pm1.87}$	88.63$_{\pm1.51}$	92.38$_{\pm0.88}$	50%	16.6%	16.6%
	4	92.01$_{\pm1.92}$	88.07$_{\pm1.69}$	92.11$_{\pm1.00}$	50%	10%	10%
	8	90.95$_{\pm1.78}$	86.28$_{\pm1.70}$	91.95$_{\pm0.77}$	50%	5.55%	5.55%
	16	87.50$_{\pm1.85}$	83.51$_{\pm1.88}$	92.33$_{\pm0.98}$	50%	2.94%	2.94%
	32	85.18$_{\pm2.01}$	82.12$_{\pm1.74}$	91.86$_{\pm1.12}$	50%	1.51%	1.51%

Table 5: PGAC's performance and corresponding data setup. **ori. acc**: main task accuracy. **# pa.**: number of benign passive parties.

Class Transferable Probability. Figure 2(b) shows that the estimated transferable probabilities for shared classes (yellow bars) are higher than those for outlier classes (cyan bars). This difference highlights that PGAC prioritizes and transfers relevant knowledge by assigning higher probabilities to shared classes and filtering out outlier classes.

Visualization. Figure 3 presents the t-SNE visualization of learned embeddings after adaptation. Key observations include: (i) PGAC generates more generalized embeddings, showing that samples are dispersedly distributed in the space. (ii) PGAC effectively separates each class of samples in both the source and target domains. (iii) PGAC correctly aligns the source and target embeddings within shared classes.

Ablation Study. We assess components in PGAC, including *Generalized Embedding Learning* (GEL), *Domain Knowledge Transfer* (DKT), and *Shadow Input Construction* (SIC), by designing PGAC variants to test performances. As shown

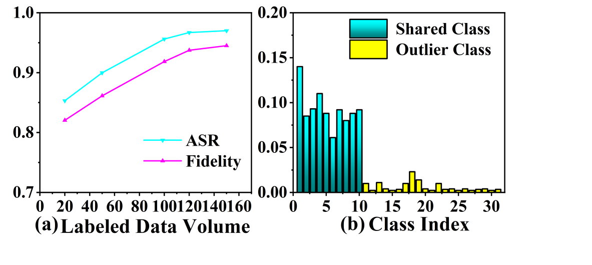

Figure 2: (a) Impact of labeled data volume. (b) Estimated class transferable probability $\mathbf{w}$.

in Table 6, all components contribute to the attacks.

4.5 Results against Countermeasures

We investigate three potential countermeasures: ***(i) Noisy Gradients***. Adds Laplacian noise to the gradients before shar-

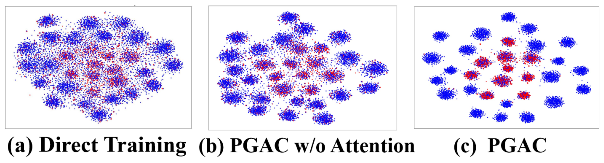

Figure 3: Visualization of source (blue) and target embeddings (red).

GEL	DKT	SIC	ASR	Fidelity
✓	✓	✓	95.60 ± 1.13	91.84 ± 0.77
✗	✗	✗	52.71 ± 1.21	42.11 ± 1.98
✓	✗	✗	65.85 ± 1.70	60.84 ± 0.92
✓	✓	✗	83.77 ± 1.20	91.84 ± 0.77

Table 6: Performance of PGAC and its variants.

ing them with parties to obscure gradient information and reduce the embedding model's expressiveness. ***(ii) Gradient Compression***. Shares only a subset of gradients that have the largest absolute values to limit the amount of gradient information available. ***(iii) Private Training***. During each iteration, selects a single gradient, adds noise, and retains it only if the noisy gradient exceeds a predefined threshold τ. This process continues until a specific fraction θ_u of gradient values is retained. We evaluate defense methods against PGAC under both configurations on the Office-31 dataset.

Results. ***(i) Noisy Gradients.*** Figure 4(a) reveals that low levels of noise in gradients have minimal effect in mitigating PGAC. However, adding sufficient noise scale may flip the signs of gradients and counter PGAC. ***(ii) Gradient Compression.*** Figure 4(b) shows that gradient compression does not significantly deter PGAC's effectiveness. This is because it typically removes smaller gradients, which may not include those related to labels. ***(iii) Private Training.*** Figure 4(c) shows that setting θ_u to 0.25 or lower counters PGAC. However, this stringent setting reduces classification accuracy for the main task to 61.38%.

5 Related Work

Conventional Adversarial Attacks. One type of attack is to enhance the transferability of AEs: Momentum-based approaches improve the stability of update directions during iterations, avoiding poor local maxima. MIFGSM [Dong *et al.*, 2018] first incorporated momentum into iterative attacks, followed by enhancements such as adaptive momentum variance [Li *et al.*, 2023b], variance tuning [Wang and He, 2021], multi-step gradient accumulation [Jang *et al.*, 2022], and exploring gradient relevance between inputs and their neighborhoods [Zhu *et al.*, 2023]. Input transformation-based approaches apply transformations to the input. Techniques include random patch transformations [Zhang *et al.*, 2022], pasting grid masks on the original image [Hong *et al.*, 2022], merging with translated versions of the same or other images [Wang *et al.*, 2021], scaling different areas of the input image at varying percentages (e.g., SINIFGSM [Lin *et al.*, 2020]), randomly transforming image blocks [Wang *et al.*, 2023b], and applying combined feature enhancement transformations on clean image copies during iterative updates [Ren *et al.*, 2023]. Ensemble-based approaches attack multiple models concurrently. [Liu *et al.*, 2017] confirmed the effectiveness of this strategy, with works like DIFGSM [Xie *et al.*, 2018], and [Wang and He, 2021; Xiong *et al.*, 2022] solidifying its advantages. However, this type of method requires access to pre-trained models from libraries, which may not be available in many real-world tasks. Another type focuses on training a proxy model that closely approximates the victim model [Papernot *et al.*, 2017; Orekondy *et al.*, 2019], enabling crafted AEs to transfer naturally to the victim model. However, this approximation training relies on real labeled training data, which is hard to acquire due to privacy or transmission reasons.

Adversarial Attacks in VFL. There is limited literature on adversarial attacks against VFL. ADI was proposed to perturb attacker-controlled features to explicitly diminish other parties' input contributions and dominate the VFL's prediction [Pang *et al.*, 2022]. However, it relies on querying the server model and accessing test input features from other parties. In [Qiu *et al.*, 2022], zeroth-order optimization (ZOO) was used to craft optimal perturbations on controlled embeddings, while a poisoning process during VFL training was applied to amplify the embedding's influence on prediction by enforcing associations with the target class. However, it requires labeled training data to build a proxy model. In this paper, we propose PGAC, which operates exclusively during the joint inference phase, using auxiliary images from non-training domains to craft adversarial inputs.

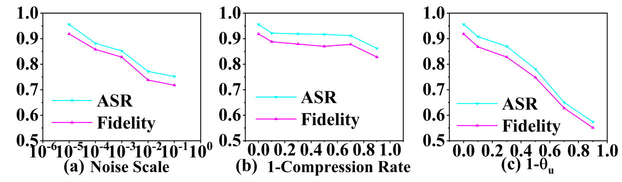

Figure 4: Defense performance of three methods: (a) noisy gradients, (b) gradient compression, and (c) private training.

6 Conclusion

In this paper, we proposed PGAC, the first adversarial attack framework against VFL under a more relaxed setting. Unlike prior attacks, PGAC does not rely on knowledge of the server model, real test inputs from other parties, and auxiliary data from the training domain. PGAC generates a high-fidelity proxy for joint inference and a diverse set of salient contribution shadow inputs. This simulation enables the crafting of highly transferable contribution-monopoly adversarial inputs, thereby hogging the rewards provided to incentivize VFL parties contributing important features. Extensive experiments demonstrate that PGAC achieves notable performance across various settings and significantly outperforms state-of-the-art methods within the same threat model.

Acknowledgments

This work is supported by NSFC (Grant No. 62272038, U2241213), Beijing Natural Science Foundation (Grant No. M23018), National Key R&D Program of China (Grant No. 2022YFB2702903).

References

[Bai *et al.*, 2023] Yijie Bai, Yanjiao Chen, Hanlei Zhang, Wenyuan Xu, Haiqin Weng, and Dou Goodman. {VILLAIN}: Backdoor attacks against vertical split learning. In *USENIX Security 23*, pages 2743–2760, 2023.

[Cao *et al.*, 2018] Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *ECCV 2018*, pages 135–150, 2018.

[Chen *et al.*, 2017] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In *CCS 2017*, pages 1243–1255, 2017.

[Ciesielski and Leszczynski, 2006] Mariusz Ciesielski and Jacek Leszczynski. Numerical treatment of an initial-boundary value problem for fractional partial differential equations. *Signal Process.*, 86(10):2619–2631, 2006.

[Dong *et al.*, 2018] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *CVPR 2018*, pages 9185–9193, 2018.

[Fu *et al.*, 2022] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X. Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *USENIX Security 2022*, pages 1397–1414, 2022.

[Ganin *et al.*, 2016] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

[Goodfellow *et al.*, 2015] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR 2015*, 2015.

[Griffin *et al.*, 2007] Gregory Griffin, Alex Holub, Pietro Perona, et al. Caltech-256 object category dataset. Technical report, Technical Report 7694, California Institute of Technology Pasadena, 2007.

[Gu *et al.*, 2021] Xiang Gu, Xi Yu, Yan Yang, Jian Sun, and Zongben Xu. Adversarial reweighting for partial domain adaptation. In *NeurIPS 2021*, pages 14860–14872, 2021.

[Hard *et al.*, 2018] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604, 2018.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR 2016*, pages 770–778, 2016.

[Hong *et al.*, 2022] Jinbang Hong, Keke Tang, Chao Gao, Songxin Wang, Sensen Guo, and Peican Zhu. Gm-attack: Improving the transferability of adversarial attacks. In *KSEM 2022*, volume 13370, pages 489–500, 2022.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR 2017*, pages 4700–4708, 2017.

[Jang *et al.*, 2022] Donggon Jang, Sanghyeok Son, and Dae-Shik Kim. Strengthening the transferability of adversarial examples using advanced looking ahead and self-cutmix. In *CVPR Workshops 2022*, pages 147–154, 2022.

[Li *et al.*, 2023a] Anran Li, Hongyi Peng, Lan Zhang, Jiahui Huang, Qing Guo, Han Yu, and Yang Liu. Fedsdg-fs: Efficient and secure feature selection for vertical federated learning. In *INFOCOM 2023*, pages 1–10, 2023.

[Li *et al.*, 2023b] Chao Li, Wen Yao, Handing Wang, and Tingsong Jiang. Adaptive momentum variance for attention-guided sparse adversarial attacks. *Pattern Recognit.*, 133:108979, 2023.

[Lin *et al.*, 2020] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E. Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *ICLR 2020*, 2020.

[Liu *et al.*, 2017] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR 2017*, 2017.

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

[Long *et al.*, 2018] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional adversarial domain adaptation. In *NeurIPS 2018*, pages 1647–1657, 2018.

[Madry *et al.*, 2017] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017.

[Mammen, 2021] Priyanka Mary Mammen. Federated learning: Opportunities and challenges. *arXiv preprint arXiv:2101.05428*, 2021.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS 2017*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282, 2017.

[Orekondy *et al.*, 2019] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *CVPR 2019*, pages 4954–4963, 2019.

[Pang *et al.*, 2022] Qi Pang, Yuanyuan Yuan, Shuai Wang, and Wenting Zheng. Adi: Adversarial dominating inputs in vertical federated learning systems. *arXiv preprint arXiv:2201.02775*, 2022.

[Papernot *et al.*, 2017] Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *AsiaCCS 2017*, pages 506–519, 2017.

[Peng *et al.*, 2019] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV 2019*, pages 1406–1415, 2019.

[Poirot *et al.*, 2019] Maarten G Poirot, Praneeth Vepakomma, Ken Chang, Jayashree Kalpathy-Cramer, Rajiv Gupta, and Ramesh Raskar. Split learning for collaborative deep learning in healthcare. *arXiv preprint arXiv:1912.12115*, 2019.

[Qiu *et al.*, 2022] Pengyu Qiu, Xuhong Zhang, Shouling Ji, Changjiang Li, Yuwen Pu, Xing Yang, and Ting Wang. Hijack vertical federated learning models with adversarial embedding. *CoRR*, abs/2212.00322, 2022.

[Ren *et al.*, 2023] Yuchen Ren, Hegui Zhu, Xiaoyan Sui, and Chong Liu. Crafting transferable adversarial examples via contaminating the salient feature variance. *Inf. Sci.*, 644:119273, 2023.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[Saenko *et al.*, 2010] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 213–226, 2010.

[Saito *et al.*, 2017] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *ICML 2017*, pages 2988–2997, 2017.

[Saito *et al.*, 2019] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised domain adaptation via minimax entropy. In *ICCV 2019*, pages 8050–8058, 2019.

[Selvaraju *et al.*, 2017] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV 2017*, pages 618–626, 2017.

[Shi *et al.*, 2022] Cheng Shi, Yenan Dang, Li Fang, Minghua Zhao, Zhiyong Lv, Qiguang Miao, and Chi-Man Pun. Multifeature collaborative adversarial attack in multimodal remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–15, 2022.

[Sim *et al.*, 2020] Rachael Hwee Ling Sim, Yehong Zhang, Mun Choon Chan, and Bryan Kian Hsiang Low. Collaborative machine learning with incentive-aware model rewards. In *ICML 2020*, pages 8927–8936, 2020.

[Simonyan, 2014] Karen Simonyan. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Wang and He, 2021] Xiaosen Wang and Kun He. Enhancing the transferability of adversarial attacks through variance tuning. In *CVPR 2021*, pages 1924–1933, 2021.

[Wang *et al.*, 2021] Xiaosen Wang, Xuanran He, Jingdong Wang, and Kun He. Admix: Enhancing the transferability of adversarial attacks. In *ICCV 2021*, pages 16138–16147, 2021.

[Wang *et al.*, 2023a] Shuo Wang, Keke Gai, Jing Yu, and Liehuang Zhu. Vfedmh: Vertical federated learning for training multi-party heterogeneous models. *CoRR*, abs/2310.13367, 2023.

[Wang *et al.*, 2023b] Xiaosen Wang, Zeliang Zhang, and Jianping Zhang. Structure invariant transformation for better adversarial transferability. In *ICCV 2023*, pages 4584–4596, 2023.

[Xie *et al.*, 2018] Cihang Xie, Zhishuai Zhang, Jianyu Wang, Yuyin Zhou, Zhou Ren, and Alan L. Yuille. Improving transferability of adversarial examples with input diversity. *CoRR*, abs/1803.06978, 2018.

[Xie *et al.*, 2020] Qizhe Xie, Minh-Thang Luong, Eduard H. Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification. In *CVPR 2020*, pages 10684–10695, 2020.

[Xiong *et al.*, 2022] Yifeng Xiong, Jiadong Lin, Min Zhang, John E. Hopcroft, and Kun He. Stochastic variance reduced ensemble adversarial attack for boosting the adversarial transferability. In *CVPR 2022*, pages 14963–14972, 2022.

[Zhang *et al.*, 2022] Yaoyuan Zhang, Yu-an Tan, Tian Chen, Xinrui Liu, Quanxin Zhang, and Yuanzhang Li. Enhancing the transferability of adversarial examples with random patch. In *IJCAI 2022*, pages 1672–1678, 2022.

[Zhou *et al.*, 2020] Mingyi Zhou, Jing Wu, Yipeng Liu, Shuaicheng Liu, and Ce Zhu. Dast: Data-free substitute training for adversarial attacks. In *CVPR 2020*, pages 234–243, 2020.

[Zhu *et al.*, 2023] Hegui Zhu, Yuchen Ren, Xiaoyan Sui, Lianping Yang, and Wuming Jiang. Boosting adversarial transferability via gradient relevance attack. In *ICCV 2023*, pages 4718–4727, 2023.