Generic Adversarial Attack Framework Against Graph-based Vertical Federated Learning

Yimin Liu[1], **Peng Jiang**[1*], **Qi Liu**[1], **Liehuang Zhu**[1]

[1]School of Cyberspace Science and Technology, Beijing Institute of Technology, China
{3120256074, pengjiang, 3120256077, liehuangz}@bit.edu.cn

Abstract

Graph-based vertical federated learning (GVFL) enables multiple parties to collaboratively train and infer over aligned nodes, where each party contributes its own local embedding derived from different attributes and adjacency relations. Adversarial inputs injected by an attacker can skew the joint prediction toward its desired outcomes while diminishing the influence of benign parties and undermining contribution. However, most attacks typically have pre-set assumptions, such as access to the server architecture, model queries, or in-domain auxiliary graphs. In this paper, we propose SGAC, an attack framework that enables domination of joint inference without relying on above assumptions. SGAC learns label-indicative embeddings and class-transferable probabilities to generate a surrogate that closely mimics the server-side classification behavior by exploiting auxiliary graphs from non-training domains. SGAC then leverages saliency over node attributes and edges on the auxiliary graphs to construct a diverse set of shadow inputs resembling highly influential test instances. With the surrogate fidelity and input diversity, SGAC crafts transferable contribution-monopoly adversarial inputs that hijack GVFL incentives. Extensive experiments across diverse model architectures validate SGAC's effectiveness.

1 Introduction

Federated learning (FL) enables multiple parties to collaboratively train models without disclosing private data, which has been widely deployed in cross-silo applications (McMahan et al. 2017). According to data distribution modes among parties, FL can be categorized into horizontal federated learning (HFL) and vertical federated learning (VFL) (Hard et al. 2018). HFL involves parties with different data samples but the same feature space, while VFL applies to parties with the same samples but complementary feature spaces (Wang et al. 2023). As graph-structured data becomes widely available, attributes and adjacency relations of the same node are often fragmented across silos, driving the need for global analysis on node-level tasks. Graph-based VFL (GVFL) offers the unique advantage of incorporating the scattered data from aligned nodes (Ni et al. 2021), and has gained prominence in real-world node classification tasks such as e-commerce platforms, traffic forecasting, and social network analysis (Fu et al. 2022b). GVFL involves collaboration between the *active party* and *passive parties*. The active party owns labeled nodes within its graph and trains a *server model* for node classification, while the passive parties possess additional attributes and adjacency edges corresponding to the same set of nodes (Fu et al. 2022a). Instead of sharing raw graph data, each party generates input embedding locally using its own *embedding model*, which is then aggregated into the server model for joint training and subsequent inference (Bai et al. 2023).

Although GVFL is designed to integrate all parties' inputs into decision-making, it remains vulnerable to adversarial inputs (Pang et al. 2022; Chen et al. 2023). A passive party that turns adversarial can slightly perturb its input to mislead joint inference. Such an injection enables the attacker to drive predictions toward a target class and monopolize rewards by making benign contributions negligible (Sim et al. 2020). Despite recent advances, most methods still rely on pre-set assumptions. For example, Graph-Fraudster requires access to the server model's architecture and independently issues queries for prediction and gradient feedback (Chen et al. 2023). In GVFL, the server model is owned by the active party and its construction encodes proprietary expertise, making it inaccessible in practice. Moreover, such resource-intensive and solely party-side queries are costly and trigger suspicion. Other works such as NA^2 and Hijack (Chen, Zhang, and Zheng 2024; Qiu et al. 2022) rely on labeled data sharing the same data distribution and class space as the training data. This is not consistent with real-world graphs, which usually exhibit cross-domain disparities.

In this paper, we propose SGAC, a generic adversarial attack framework against GVFL. SGAC enables a passive party to craft adversarial inputs that dominate joint inference using labeled auxiliary graph data that differs from the training data in both distribution and class space. SGAC consists of two modules: **S**urrogate **G**eneration and **A**dversarial **C**rafting. The surrogate generation module learns a high-fidelity surrogate of the server model by employing a multi-task discriminator with class-conditioned distribution alignment and class-wise weighting. This discriminator assigns instances to appropriate alignment tasks in the shared class space and filters out irrelevant data from mismatched or outlier classes. The adversarial crafting module constructs a diverse, contribution-emphasized shadow input set to guide

*Corresponding author.

perturbation optimization and suppress influential test embeddings in surrogate joint inference. It iteratively optimizes perturbations so that the attacker's embedding dominates joint inference while other parties' inputs have negligible influence. With both the surrogate's high fidelity and the diversity of shadow inputs, the crafted adversarial inputs transfer and dominate the server model's joint inference. Finally, we evaluate SGAC under three potential defenses.

The main contributions of this paper are twofold.

- We propose SGAC, an adversarial attack framework for GVFL that operates without server access or in-domain labeled data. SGAC produces a high-fidelity surrogate model and influential shadow inputs, enabling the attacker's inputs to transferably dominate joint inference.
- Extensive experiments confirm the effectiveness of SGAC, achieving an 87.38% attack success rate and outperforming state-of-the-art attacks by at least 24.84%. Results under three countermeasures further show that SGAC generally remains effective against the defenses.

2 Problem Statement

2.1 GVFL Training and Inference

Consider a node classification task with m parties $\{\mathcal{P}^1, \ldots, \mathcal{P}^m\}$, each $\mathcal{P}^i$ holding a personalized training graph $\mathbf{G}^i_{\text{train}} = (\mathcal{V}_{\text{train}}, X^i_{\text{train}}, A^i_{\text{train}})$ and test graph $\mathbf{G}^i_{\text{test}} = (\mathcal{V}_{\text{test}}, X^i_{\text{test}}, A^i_{\text{test}})$. Here $\mathcal{V} \in \{\mathcal{V}_{\text{train}}, \mathcal{V}_{\text{test}}\}$ is the set of common nodes[1] and $X^i \in \mathbb{R}^{|\mathcal{V}| \times d^i}$ is the node attribute (i.e., feature) matrix, $A^i \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ is the adjacency matrix.

In GVFL, 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. 2022a). Each party $\mathcal{P}^i$ trains a local embedding model, i.e., a Graph Neural Network (GNN), to embed each training node u_{tra} into a latent embedding $\mathbf{h}_{u_{\text{tra}}}$ by aggregating node attributes with neighbors in $\mathbf{G}^i_{\text{train}}$: $\mathbf{h}^i_{u_{\text{tra}}} = \mathcal{M}^i_e\left(\mathbf{x}^i_{u_{\text{tra}}}, \{\mathbf{x}^i_v\}_{v \in \mathcal{N}^i(u_{\text{tra}})}, A^i_{\text{train}}; \theta^i\right) \in \mathbb{R}^{d^i_*}$, where $\mathbf{x}^i_{u_{\text{tra}}}$ is the d^i-dimensional attribute vector of node u_{tra} from X^i_{train}, $\mathcal{N}^i(u_{\text{tra}})$ is the neighbors of u_{tra} from A^i_{train} (Ni et al. 2021). 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 $\mathbf{h}_{\text{cat}} = [\mathbf{h}^1_{u_{\text{tra}}}, \ldots, \mathbf{h}^m_{u_{\text{tra}}}]$. The server model maps $\mathbf{h}_{\text{cat}}$ to $\mathbf{h}_{\text{top}}$: $\mathcal{M}_s(\mathbf{h}_{\text{cat}}; \theta_{\text{top}}) : \mathbb{R}^{d^1_* + \cdots + d^m_*} \to \mathbb{R}^c$, where c is the number of classes. The output $\mathbf{h}_{\text{top}}$ is used to compute loss, e.g., cross-entropy loss, to optimize the parameters of all models. Training objective is formulated as:

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

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 $\mathcal{M}^i_e(.; \theta^i)$.

During inference, each party $\mathcal{P}^i$ computes the local embedding for each test node u_{test} in $\mathbf{G}^i_{\text{test}}$ as: $\mathbf{H}^i_{u_{\text{test}}} = \mathcal{M}^i_e\left(\mathbf{x}^i_{u_{\text{test}}}, \{\mathbf{x}^i_v\}_{v \in \mathcal{N}^i(u_{\text{test}})}, A^i_{\text{test}}; \theta^i\right)$, where $\mathbf{x}^i_{u_{\text{test}}}$ is the d^i-dimensional attribute vector of node u_{test} from X^i_{test}, and $\mathcal{N}^i(u_{\text{test}})$ are neighbors of u_{test} from A^i_{test} (Bai et al. 2023). All test input embeddings are concatenated into $\mathbf{H}_{\text{cat}}$, which is also fed into the server model to compute 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 then obtained as: $\hat{y}^s = \arg\max(p)$. For notational simplicity, we use $\mathbf{h}^i$, $\mathbf{H}^i$ denote $\mathbf{h}^i_{u_{\text{tra}} \in \mathcal{V}_{\text{train}}}$ and $\mathbf{H}^i_{u_{\text{test}} \in \mathcal{V}_{\text{test}}}$ in the following.

2.2 Adversarial Attacks in GVFL

Following prior work, we assume that the m parties have aligned their common nodes beforehand (Bai et al. 2023; Fu et al. 2022a; Pang et al. 2022). Among the remaining $m-1$ passive parties, one is the attacker $\mathcal{A}$. $\mathcal{A}$ follows the normal protocol but attempts to craft adversarial input embedding that dominates the server model's joint inference toward a desired class l_{tar} (Chen et al. 2023; Qiu et al. 2022).

Attacker's Capability & Knowledge. We assume that $\mathcal{A}$ can train a proxy version of server model $\mathcal{M}_s$ with auxiliary data and adversarially perturb its controlled test input $\mathbf{H}^{\mathcal{A}}$.

$\mathcal{A}$ lacks access to $\mathcal{M}_s$ (including the structure, parameters and query-based predictions), other parties' test inputs $\mathbf{H}^{\mathcal{B}}$ and embedding model $\mathcal{M}^{\mathcal{B}}_e$, and any labeled training data. $\mathcal{A}$ has its training graph $\mathbf{G}^{\mathcal{A}}_{\text{train}}$ and test graph $\mathbf{G}^{\mathcal{A}}_{\text{test}}$, optimized embedding model $\mathcal{M}^{\mathcal{A}}_e$ (including shared gradients $g^{\mathcal{A}}$) and access to a labeled auxiliary graph $\mathbf{G}_{\text{aux}} = (X_{\text{aux}} \in \mathbb{R}^{N_{\text{aux}} \times d_{\text{aux}}}, A_{\text{aux}} \in \{0,1\}^{N_{\text{aux}} \times N_{\text{aux}}})$ from non-training domain. $\mathbf{G}_{\text{aux}}$ differs in data distribution from $\mathbf{G}^{\mathcal{A}}_{\text{train}}$ and encompasses $\mathbf{G}^{\mathcal{A}}_{\text{train}}$'s class space $\mathcal{C}_t$ within its broader class space $\mathcal{C}_s$. The exact overlap of class spaces between $\mathbf{G}_{\text{aux}}$ and $\mathbf{G}^{\mathcal{A}}_{\text{train}}$ (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 graph includes specific disease categories, auxiliary graphs could come from larger medical institutions to encompass a wider range of diseases. $\mathcal{A}$ is aware that class l_{tar} is included in $\mathcal{C}$ (Pang et al. 2022).

Definition of Adversarial Attacks. Conventional attacks can directly craft an attacker-controlled full test input $\mathbf{H}^{\mathcal{A}}$ to be misclassified into a desired class l_{tar} by solving the following optimization problem (Zhou et al. 2020):

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

where f is the victim classifier, δ is the perturbation.

In GVFL, each party holds only a subset of each test node's attributes and adjacency edges. Let $\mathbf{H}^{\mathcal{B}} = [\mathbf{H}^{\mathcal{B}_1}, \ldots, \mathbf{H}^{\mathcal{B}_{m-1}}]$ denote the input embeddings from all benign parties. During joint inference, the attacker-controlled input $\mathbf{H}^{\mathcal{A}}$ and the benign inputs $\mathbf{H}^{\mathcal{B}}$ are aggregated into the server model $\mathcal{M}_s$ to produce predictions. To dominate joint inference toward l_{tar} regardless of any benign input contribution, perturbations are optimized to minimize the loss for l_{tar} while penalizing saliency of benign inputs (Pang et al. 2022), formulated as:

$$\min_{\delta} \left[\ell_{ce}\left(\mathcal{M}_s(\mathbf{H}^{\mathcal{A}} + \delta, \mathbf{H}^{\mathcal{B}}), l_{tar}\right) + \alpha \cdot \textbf{\textit{Saliency}}(\mathbf{H}^{\mathcal{B}})\right].$$

[1] Before training and inference, GVFL determines $\mathcal{V}$ as the intersection of local node sets across parties (Chen, Laine, and Rindal 2017), and aligns attributes, adjacency relations per common node.

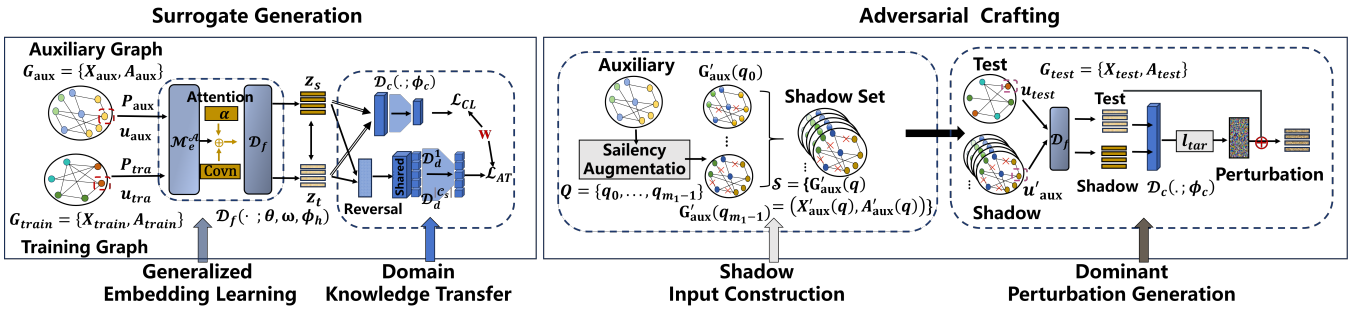

Figure 1: Overview of SGAC.

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

$$\textit{Saliency}(\mathbf{H}^{\mathcal{B}}) = \left\| \frac{\partial}{\partial \mathbf{H}^{\mathcal{B}}} \mathrm{Var}(\textit{output}) \right\|_1 .$$

3 Methodology

As shown in Figure 1, SGAC consists of *Surrogate Generation* that learns a high-fidelity surrogate for gradient-based targeted perturbations and *Adversarial Crafting* that constructs diverse contribution-emphasized shadow inputs to iteratively produce dominant adversarial perturbations.

3.1 Surrogate Generation

Given the source (i.e., auxiliary) graph $\mathbf{G}_{\text{aux}}$ and the target (i.e., training) graph $\mathbf{G}^{\mathcal{A}}_{\text{train}}$, we extract transferable features that mitigate domain shift, inspired by domain-adversarial networks (Ganin et al. 2016). 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 surrogate model $\mathcal{D}_c(.;\phi_c)$ is trained to categorize nodes.

Generalized Embedding Learning. Considering that relying on local graph consistency (i.e., direct neighbor connections) and encoder parameters alone for domain-invariant embedding learning may inadequately shift distributions, we explore high-order co-occurrence patterns in graphs and incorporate an attention layer into the encoder $\mathcal{D}_f$. This design enables $\mathcal{D}_f$ to capture topological consistency, label-indicative features from raw graph and learn generalized embedding to promote better alignment across domains.

Given the graph $\mathbf{G} \in \{\mathbf{G}_{\text{aux}}, \mathbf{G}^{\mathcal{A}}_{\text{train}}\}$, we perform random walks on each adjacency matrix $A \in \{A_{\text{aux}}, A_{\text{train}}\}$ to sample a set of paths, constructing a co-occurrence frequency matrix $F \in \mathbb{R}^{N\times N}$, where $N \in \{N_{\text{aux}}, N_{\text{train}}\}$ is the node count. Each entry F_{ij} records how often node n_j appears within a fixed window around node n_i. The positive mutual information for A is calculated by:

$$P_{ij} = \max\left\{\log \frac{F_{ij}/\sum F}{(\sum_j F_{ij}/\sum F)\,(\sum_i F_{ij}/\sum F)}\,,\ 0\right\},$$

where $F_{ij}/\sum F$ is the probability of node n_j appearing in node n_i's context. $\sum_j F_{ij}/\sum F$ and $\sum_i F_{ij}/\sum F$ are the probabilities of node n_i as the anchor and node n_j as the context, respectively. P_{ij} is the positive mutual information between n_i and n_j, reflecting the high-order topological proximity between nodes. We replace the original adjacency matrix A with P, and input it alongside the attribute matrix $X \in \{X_{\text{aux}}, X_{\text{train}}\}$ into $\mathcal{D}_f$ to generate node embeddings.

Formally, the d_*-dimensional embedding $\mathbf{h} \in \{\mathbf{h}_s, \mathbf{h}_t\}$ is produced by $\mathcal{D}_f$, where $\mathbf{h}_s$ and $\mathbf{h}_t$ are the embeddings of each source node u_{aux} in $\mathbf{G}_{\text{aux}}$ and each target node u_{tra} in $\mathbf{G}^{\mathcal{A}}_{\text{train}}$, respectively. To characterize 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 embedding $\mathbf{z}$ is then obtained by applying a convolution operation:

$$\mathbf{z} = \mathrm{Conv}([\hat{\mathbf{h}}^1, \ldots, \hat{\mathbf{h}}^{|\mathcal{C}_s|}]; \phi_h) \in \mathbb{R}^{d_*},$$

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 embeddings $\mathbf{z} \in \{\mathbf{z}_s, \mathbf{z}_t\}$, 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 surrogate model.

Domain Knowledge Transfer. We adopt a multi-task domain discriminator $\mathcal{D}_d$ with shared bottom layers and $|\mathcal{C}_s|$ class-specific heads $\{\mathcal{D}^{(k)}_d\}^{|\mathcal{C}_s|}_{k=1}$ to mitigate negative transfer caused by aligning the entire source and target distributions when the overlap between $\mathcal{C}_s$ and $\mathcal{C}_t$ is unknown. 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 the surrogate model, $\mathbf{z}$ is stacked as a double input to simulate an attacker–vs.–proxy-benign surrogate, mapping $\mathbb{R}^{2\times d_*} \to \mathbb{R}^{|\mathcal{C}_s|}$. This $\mathcal{D}_c$ modeling is consistent with the joint contributions of the attacker-controlled and benign inputs to the server model (Fu et al. 2022a; Bai et al. 2023). Since output $\hat{p}$ characterizes the probability of assigning $\mathbf{z}$ to each of the $|\mathcal{C}_s|$ source classes (Saito, Ushiku, and Harada

2017), we embed $\hat{p}$ as an instance weight in the domain discriminator loss for all $|\mathcal{C}_s|$ heads. We then estimate a class-level transferability weight for each source class and use it to down-weight the domain discriminator heads associated with outlier source classes $k \in \bar{\mathcal{C}}_s$, whose $\mathcal{D}_d^{(k)}$ provide irrelevant alignment signals. Since target embeddings $\mathbf{z}_t$ are unlikely to belong to these outlier classes, we define $\mathbf{w} = \mathbb{E}_{\mathbf{z}_t \sim \mathcal{Z}_t}\big[\mathcal{D}_c(\mathbf{z}_t, \mathbf{z}_t; \phi_c)\big]$, so that classes rarely activated by target data receive smaller weights. Following the entropy-based weighting (Long et al. 2018), we further emphasize confidently predicted (thus more transferable) instances with $w_e(\mathbf{z}) = 1 + e^{-J(\mathcal{D}_c(\mathbf{z},\mathbf{z};\phi_c))}$, where J denotes the entropy function. The final discriminator loss is:

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

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).

$\mathcal{L}_{AT}$ builds on the surrogate classifier $\mathcal{D}_c$ over both source and target domains. We incorporate the class-transferability weights $\mathbf{w}$ into the supervised loss of $\mathcal{D}_c$ so that source samples from likely shared classes are emphasized and outlier classes are down-weighted. For unlabeled target embeddings, we apply a self-training loss with pseudo labels $\hat{y}^c_{\text{self}} = \arg\max(\mathcal{D}_c(\mathbf{z}_t, \mathbf{z}_t; \phi_c))$, again weighted by $\mathbf{w}$ to confine predictions to the shared class space (Xie et al. 2020). Considering that attackers retain gradients for their local data during GVFL training, we introduce a gradient-similarity loss ℓ_2 to enforce $\mathcal{D}_c$'s fidelity to the server model by minimizing the discrepancy between the retained gradient $g_t^{\mathcal{A}}$ and surrogate gradient g_t on attacker's local embedding $\mathbf{h}_t$, formulated as: $\ell_2(g_t^{\mathcal{A}}, g_t) = \big\|g_t^{\mathcal{A}} - \partial\ell_{\text{ce}}(\mathcal{D}_c(\mathbf{z}_t, \mathbf{z}_t; \phi_c), \hat{y}^c_{\text{self}})/\partial\mathbf{h}_t\big\|_2$. Thus, the surrogate model loss $\mathcal{L}_{CL}$ is formulated as:

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

Optimization. Integrating all losses, the objective of surrogate 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 the auxiliary graph $\mathbf{G}_{\text{aux}} = (X_{\text{aux}}, A_{\text{aux}})$, we simulate a latent party $\mathcal{B}'$ holding influential test inputs during joint inference. Specifically, entries in X_{aux} and A_{aux} that contribute to classification are emphasized, then augmented to construct a diverse shadow graph set $\mathcal{S}$ for $\mathcal{B}'$. Building on the surrogate model $\mathcal{D}_c$ and $\mathcal{S}$, adversarial perturbations are iteratively added to $\mathbf{H}^{\mathcal{A}}$, optimized to diminish the contributions of influential inputs to the joint inference.

Shadow Input Construction. We begin by capturing discriminative features for each node n (i.e., u_{aux}) in $\mathbf{G}_{\text{aux}}$. Let $\mathbf{x}_n \in \mathbb{R}^{d_{\text{aux}}}$ denote the raw feature vector from X_{aux}, and y_n^c the logit output of surrogate model $\mathcal{D}_c$ for node n's true class c. Inspired by Grad-CAM (Selvaraju et al. 2017), we quantify each feature's contribution to the model's prediction:

$$L_{n,p}^c = \text{ReLU}\big(\alpha_{n,p}^c \cdot \mathbf{x}_{n,p}\big),$$

where $\alpha_{n,p}^c = \partial\, y_n^c / \partial\, \mathbf{x}_{n,p}$ is the dimension-wise saliency weight. To emphasize salient node features, entries below the q-th percentile of L_n^c are suppressed as:

$$C_n^c(q) = \big[L_{n,p}^c \cdot \mathbb{I}(L_{n,p}^c > P_q(L_n^c))\big]_{p=1}^{d_{\text{aux}}},$$

where $P_q(L_n^c)$ is the q-th percentile of the d_{aux} entries of L_n^c, and $\mathbb{I}(\cdot)$ is the indicator. The masked saliency vector $C_n^c(q)$ is then fused with the original node feature $\mathbf{x}_n$ to produce the contribution-emphasized feature vector $\mathbf{x}'_n(q)$.

$$\mathbf{x}'_n(q) = (1-\mu)\,\mathbf{x}_n \;+\; \mu\, C_n^c(q) \;+\; \xi,$$

where $\mu \in [0,1]$ and $\xi \sim \mathcal{N}(0, \sigma^2)$ is random noise for diversity. Since $C_n^c(q)$ is derived from the $\mathbf{x}_n$, the original label of node n remains unchanged.

After emphasizing each node's feature into $X'_{\text{aux}}(q)$, we compute each node's importance score via normalization:

$$\mathcal{J}_n \;=\; \tfrac{1}{d_{\text{aux}}} \textstyle\sum_{p=1}^{d_{\text{aux}}} \frac{L_{n,p}^c - \min_r L_{n,r}^c}{\max_r L_{n,r}^c - \min_r L_{n,r}^c} \;\in [0,1].$$

Each edge's structural contribution can then naturally be qualified by averaging its endpoints' node importances (Wei et al. 2023): $\mathcal{E}_{ij} \;=\; (\mathcal{J}_i + \mathcal{J}_j)/2$, with $(i,j) \in A_{\text{aux}}$. Similarly, to retain only the discriminative edges, we compute a binary mask and apply it to the adjacency matrix:

$$M_{ij}(q) = \mathbb{I}\big(\mathcal{E}_{ij} > P_q(\{\mathcal{E}\})\big), \quad A'_{\text{aux}}(q) = A_{\text{aux}} \odot M(q),$$

where $P_q(\{\mathcal{E}\})$ is the q-th percentile of all edge scores $\mathcal{E}_{ij}$.

To further enhance the transferability of crafted adversarial input, we choose a set of percentile thresholds:

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

where q is the percentile threshold, ϵ controls granularity, and the shadow input set $\mathcal{S}$ can be constructed as:

$$\mathcal{S} \;=\; \{\, \mathbf{G}'_{\text{aux}}(q) = (X'_{\text{aux}}(q), A'_{\text{aux}}(q)) \mid q \in Q \,\}.$$

Dominant Perturbation Generation. With $\mathcal{D}_c$ and shadow set $\mathcal{S}$, we optimize perturbations on the attacker's embedding using gradient-based updates to dominate joint inference while penalizing the saliency (Definition 1) of other parties' inputs. Since $\mathcal{A}$ cannot access real embedding models, benign-party saliency is approximated on the surrogate via FDM (Ciesielski and Leszczynski 2006), which computes the derivative of output variance w.r.t. $\mathcal{S}$. Algorithm 1 summarizes a PGD-style procedure with box constraints and momentum (Dong et al. 2018). For each shadow graph, computing node saliency over N_{aux} nodes and d_{aux} features and processing $|A_{\text{aux}}|$ edges costs $O(N_{\text{aux}} d_{\text{aux}} + |A_{\text{aux}}|)$. PGD-based optimization performs T iterations over $|Q|$ shadows, each involving forward–backward passes on d_*-dimensional embeddings across $|\mathcal{C}_s|$ classes, giving $O(T|Q|(d_* + |\mathcal{C}_s|d_*))$. Thus, the overall complexity is $O\big(|Q|\,(N_{\text{aux}} d_{\text{aux}} + |A_{\text{aux}}|) + T|Q|\,(d_* + |\mathcal{C}_s|\,d_*)\big)$.

4 Evaluation

4.1 Setting

Datasets & GVFL Configuration. We evaluate SGAC on three datasets with domain shifts: *Ali-CVR* is an e-commerce

Algorithm 1: Adversarial Input Crafting

function SALIENCY_EST($\mathbf{H}^{\mathcal{A}}, \mathbf{H}^{\mathcal{B}'}, \mathcal{D}_c$)
  $output \leftarrow \mathcal{D}_c(\mathbf{H}^{\mathcal{A}}, \mathbf{H}^{\mathcal{B}'})$
  $output_{\delta_B} \leftarrow \mathcal{D}_c(\mathbf{H}^{\mathcal{A}}, \mathbf{H}^{\mathcal{B}'} + \delta_B)$
  $saliency \leftarrow \big(\text{Var}(output_{\delta_B}) - \text{Var}(output)\big)/\delta_B$
    ▷ Var(·): variance; δ_B: a small perturbation.
  return $saliency$
end function
function AI_CRAFTING($\mathbf{H}^{\mathcal{A}}, \mathcal{D}_f, \mathcal{D}_c, l_{tar}, \mathcal{S}$)
    ▷ $\alpha, \beta, \gamma, \sigma$: weights; Λ: constraint; V: mutation.
  $V \leftarrow 0, \delta_0 \leftarrow 0, t \leftarrow 1$
  while $t \leq T$ **do**
    for each node n in $\mathbf{G}'_{\text{aux}} \in \mathcal{S}$ **do**
      $\mathbf{H}_n \leftarrow \mathcal{D}_f(\mathbf{x}'_n, \{\mathbf{x}'_v\}_{v \in \mathcal{N}(n)}, A'_{\text{aux}})$
      $\delta_t \leftarrow \arg\min_{\delta_t} \alpha \cdot \text{SALIENCY\_EST}(\mathbf{H}^{\mathcal{A}} + V + \delta_t, \mathbf{H}_n, \mathcal{D}_c)$
        $+ \beta \cdot \ell_{ce}(\mathcal{D}_c(\mathbf{H}^{\mathcal{A}} + V + \delta_t, \mathbf{H}_n), l_{tar})$
        $+ \gamma \|\delta_t\|_2, \quad \text{s.t. } \|V + \delta_t\| \leq \Lambda$
      $\delta_t \leftarrow \sigma \cdot \delta_{t-1} + \delta_t, V \leftarrow V + \delta_t, t \leftarrow t + 1$
    end for
  end while
  return $\mathbf{H}^{\mathcal{A}} + V$
end function

network with 31 product categories, collected around the shopping carnival, i.e., Ali-Normal (AN), Ali-11 (A11), Ali-Before-11 (AB11). We select the first 10 categories alphabetically to build the GVFL dataset (Guo et al. 2023). *Arxiv* is a computer-science citation network with 40 subject areas. We partition it into five time-based domains: T1 (1950–2007), T2 (1950–2009), T3 (1950–2016), T4 (2016–2018), T5 (2018–2020) following (Liu et al. 2024a), and randomly select 25 classes to build the GVFL dataset. *MAG* is a subset of the Microsoft Academic Graph, composed of six domains with 20 classes (Wang et al. 2020). We adopt four domains (China (C), Germany (G), Japan (J), and USA (U)), and use 10 randomly selected classes to build the GVFL dataset (Liu et al. 2024b). We follow the same train–test splits as in (Chen et al. 2022; Pang et al. 2022).

Our default experiments adopt the common setting with two parties, each owning about half of every node's attributes and adjacency edges within the GVFL dataset (Yao et al. 2024; Bai et al. 2023; Chen et al. 2023). We sample the raw graph dataset from domains with broader class spaces as the attacker's auxiliary dataset, selecting a default of 100 nodes per class and preserving edges among sampled nodes. Table 1 summarizes the GVFL dataset and auxiliary dataset.

Models. We adopt widely used architectures as the *embedding models*, including GAT (Velickovic et al. 2017), GCN (Kipf and Welling 2017), GraphSAGE (SAGE) (Hamilton, Ying, and Leskovec 2017). By default, GCN is used for each party. The *server model* is implemented as a two-layer MLP, while the *surrogate 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 surrogate 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, we set the percentile parameter $\epsilon = 10$ and the mix ratio $\mu = 0.6$. The crafting process uses a maximum perturbation budget of $\Lambda = 15$ and $T = 20$ iterations, following (Xie et al. 2018).

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

Baselines. We establish a Real Data benchmark, using real labeled training data for surrogate generation and test data from real parties for crafting inputs. Since state-of-the-art attacks are not directly applicable under our more relaxed assumption, we adapt them by replacing the query access in Graph-Fraudster (Chen et al. 2023) with a surrogate model trained on our cross-domain auxiliary data, and replacing the labeled training sets in NA^2 and Hijack (Chen, Zhang, and Zheng 2024; Qiu et al. 2022) with our auxiliary data while keeping their poisoning process. We also compare SGAC against gray-box methods, including SRLIM, SGAttack and HGAttack (Liu et al. 2022; Li et al. 2023b; Zhao et al. 2024).

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 SGAC approaches this benchmark with deviations of at most 12.23% in ASR and 15.60% in fidelity. SGAC consistently outperforms the state-of-the-art attacks, exhibiting improvements of at least 24.84% in ASR and 34.36% in fidelity. This superiority stems from addressing cross-domain discrepancies and mining salient segments for classification. SGAC also surpasses the gray-box graph modification methods with at least 21.80% higher ASR, highlighting the effectiveness of targeted perturbations to the intermediate embeddings uploaded during the joint inference.

4.3 Parameter Analysis

Impact of Embedding & Server Models. We evaluate SGAC's performance by varying the embedding model, server model for the active party and attacker, respectively. As shown in Tables 3 and 4, SGAC 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 Ali-CVR, with just 20 labeled nodes per class, SGAC achieves an ASR of 76.54% and fidelity of 70.36%. However, the improvement gradually stabilizes when the node number exceeds 100 per class. Notably, with only 120 labeled nodes per class, far below the GVFL dataset size, SGAC achieves an ASR of 88.54% and fidelity of 81.31%.

Impact of Party Numbers. We expand typical multi-party GVFL settings from fewer than 16 to 32 benign parties (Qiu et al. 2022; Chen et al. 2023), fixing the active party's node

Dataset	Attribute / Edge Partition		Auxiliary → GVFL (Source → Target)	Auxiliary → GVFL (Source → Target)
	Active Party	Attacker		
Ali-CVR	50%	50%	All pairwise transfers among $\{AN, A11, AB11\}$	Full 31 classes → First 10 classes
Arxiv	50%	50%	All transfers from $\{T1, T2, T3\}$ to $\{T4, T5\}$	Full 40 classes → Random 25 classes
MAG	50%	50%	All pairwise transfers among $\{C, G, J, U\}$	Full 20 classes → Random 10 classes

Table 1: GVFL configuration. Attribute/Edge partition indicates the percentage of each node's attribute dimensions and edges.

Dataset	Real Data		SGAC		Graph-Fraudster		NA[2]		Hijack		SRLIM	SGAttack	HGAttack
	ASR	Fidelity	ASR	Fidelity	ASR	Fidelity	ASR	Fidelity	ASR	Fidelity	ASR	ASR	ASR
Ali-CVR	$97.83_{\pm 0.31}$	$92.60_{\pm 0.37}$	$85.60_{\pm 0.83}$	$78.63_{\pm 0.61}$	$45.77_{\pm 1.63}$	$32.60_{\pm 1.37}$	$47.60_{\pm 1.30}$	$35.13_{\pm 1.11}$	$55.18_{\pm 1.73}$	$40.67_{\pm 2.00}$	$61.24_{\pm 2.03}$	$56.71_{\pm 1.74}$	$60.53_{\pm 1.15}$
Arxiv	$91.74_{\pm 0.23}$	$88.77_{\pm 0.22}$	$80.83_{\pm 0.37}$	$73.17_{\pm 0.40}$	$32.94_{\pm 1.04}$	$25.30_{\pm 1.13}$	$37.18_{\pm 1.40}$	$27.28_{\pm 1.50}$	$52.05_{\pm 1.30}$	$38.87_{\pm 1.01}$	$57.47_{\pm 2.01}$	$54.15_{\pm 1.19}$	$59.03_{\pm 1.20}$
MAG	$94.57_{\pm 0.38}$	$90.51_{\pm 0.14}$	$83.55_{\pm 0.50}$	$75.46_{\pm 0.49}$	$37.58_{\pm 1.70}$	$29.10_{\pm 1.28}$	$40.72_{\pm 1.13}$	$33.52_{\pm 1.15}$	$58.71_{\pm 1.05}$	$41.10_{\pm 1.22}$	$55.86_{\pm 1.43}$	$57.60_{\pm 1.07}$	$60.18_{\pm 1.14}$

Table 2: SGAC's performance compared with baselines.

$\mathcal{M}_e^{\mathcal{A}}$ \ $\mathcal{M}_e^{\mathcal{B}}$		GAT		GCN		SAGE		GIN	
	Dataset	ASR	Fidelity	ASR	Fidelity	ASR	Fidelity	ASR	Fidelity
GAT	Ali-CVR	$86.41_{\pm 1.27}$	$79.30_{\pm 1.16}$	$82.62_{\pm 0.96}$	$76.06_{\pm 0.83}$	$83.44_{\pm 1.09}$	$77.21_{\pm 1.33}$	$81.63_{\pm 1.14}$	$75.70_{\pm 1.29}$
	Arxiv	$82.45_{\pm 1.18}$	$75.41_{\pm 1.43}$	$78.36_{\pm 1.11}$	$72.43_{\pm 1.57}$	$79.21_{\pm 1.39}$	$73.91_{\pm 1.22}$	$76.75_{\pm 1.34}$	$71.25_{\pm 1.47}$
	MAG	$84.87_{\pm 1.36}$	$77.18_{\pm 1.52}$	$80.24_{\pm 1.27}$	$73.02_{\pm 1.44}$	$81.76_{\pm 1.33}$	$75.17_{\pm 1.59}$	$80.80_{\pm 1.15}$	$73.84_{\pm 1.26}$
GCN	Ali-CVR	$81.36_{\pm 1.20}$	$75.13_{\pm 1.40}$	$85.60_{\pm 0.89}$	$78.63_{\pm 0.74}$	$79.57_{\pm 1.16}$	$74.25_{\pm 1.19}$	$80.67_{\pm 0.98}$	$74.30_{\pm 1.23}$
	Arxiv	$76.75_{\pm 1.32}$	$70.40_{\pm 1.64}$	$80.83_{\pm 0.42}$	$73.17_{\pm 0.55}$	$75.12_{\pm 1.29}$	$68.71_{\pm 1.36}$	$78.05_{\pm 0.87}$	$71.43_{\pm 1.41}$
	MAG	$79.26_{\pm 1.23}$	$72.53_{\pm 1.77}$	$83.55_{\pm 0.46}$	$75.46_{\pm 0.53}$	$77.80_{\pm 1.41}$	$70.74_{\pm 1.59}$	$80.51_{\pm 1.09}$	$72.30_{\pm 1.32}$
SAGE	Ali-CVR	$81.97_{\pm 1.15}$	$76.15_{\pm 1.11}$	$84.45_{\pm 1.02}$	$78.56_{\pm 1.45}$	$87.38_{\pm 1.21}$	$80.64_{\pm 1.42}$	$84.07_{\pm 1.14}$	$77.06_{\pm 1.26}$
	Arxiv	$76.65_{\pm 1.33}$	$71.88_{\pm 1.36}$	$81.00_{\pm 1.24}$	$74.67_{\pm 1.41}$	$82.91_{\pm 1.29}$	$76.28_{\pm 1.47}$	$78.80_{\pm 1.12}$	$73.23_{\pm 1.28}$
	MAG	$78.75_{\pm 1.29}$	$72.01_{\pm 1.44}$	$84.23_{\pm 1.17}$	$77.10_{\pm 1.58}$	$83.25_{\pm 1.26}$	$77.03_{\pm 1.53}$	$80.87_{\pm 1.05}$	$75.01_{\pm 1.38}$
GIN	Ali-CVR	$81.50_{\pm 1.76}$	$75.34_{\pm 1.68}$	$80.54_{\pm 1.13}$	$74.01_{\pm 1.39}$	$78.83_{\pm 1.51}$	$73.07_{\pm 1.25}$	$84.83_{\pm 0.91}$	$77.45_{\pm 1.18}$
	Arxiv	$83.25_{\pm 1.33}$	$77.01_{\pm 1.22}$	$81.37_{\pm 1.16}$	$75.28_{\pm 1.63}$	$74.71_{\pm 1.04}$	$68.00_{\pm 0.92}$	$80.11_{\pm 1.23}$	$72.23_{\pm 1.48}$
	MAG	$80.74_{\pm 1.17}$	$73.55_{\pm 1.43}$	$78.87_{\pm 1.29}$	$71.43_{\pm 1.21}$	$76.73_{\pm 1.08}$	$70.55_{\pm 1.30}$	$82.36_{\pm 0.99}$	$74.51_{\pm 0.89}$

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

$\mathcal{D}_c$ \ $\mathcal{M}_s$		2-MLP		3-MLP		4-MLP		5-MLP		6-MLP		7-MLP	
	Dataset	ASR	Fidelity	ASR	Fidelity	ASR	Fidelity	ASR	Fidelity	ASR	Fidelity	ASR	Fidelity
2-MLP	Ali-CVR	$86.22_{\pm 1.47}$	$79.23_{\pm 1.08}$	$84.54_{\pm 1.19}$	$77.47_{\pm 1.56}$	$84.03_{\pm 1.84}$	$77.52_{\pm 1.27}$	$85.36_{\pm 1.72}$	$78.83_{\pm 1.41}$	$83.63_{\pm 1.61}$	$76.32_{\pm 1.88}$	$86.00_{\pm 1.36}$	$78.08_{\pm 1.69}$
	Arxiv	$80.63_{\pm 1.75}$	$73.14_{\pm 1.28}$	$77.35_{\pm 1.33}$	$70.14_{\pm 1.91}$	$76.36_{\pm 1.81}$	$69.72_{\pm 1.37}$	$78.63_{\pm 1.70}$	$71.50_{\pm 1.44}$	$76.77_{\pm 1.58}$	$68.45_{\pm 1.82}$	$78.60_{\pm 1.91}$	$71.51_{\pm 1.56}$
	MAG	$83.81_{\pm 1.88}$	$76.46_{\pm 1.64}$	$80.20_{\pm 1.74}$	$73.61_{\pm 1.91}$	$79.51_{\pm 1.52}$	$72.77_{\pm 1.45}$	$81.34_{\pm 1.80}$	$73.79_{\pm 1.57}$	$79.35_{\pm 1.68}$	$72.05_{\pm 1.79}$	$81.73_{\pm 1.51}$	$74.91_{\pm 1.75}$
4-MLP	Ali-CVR	$84.12_{\pm 1.92}$	$76.53_{\pm 1.68}$	$86.33_{\pm 1.77}$	$78.08_{\pm 1.81}$	$86.34_{\pm 1.41}$	$79.06_{\pm 1.73}$	$83.42_{\pm 1.88}$	$75.05_{\pm 1.69}$	$83.80_{\pm 1.56}$	$75.78_{\pm 1.60}$	$84.50_{\pm 1.74}$	$76.91_{\pm 1.66}$
	Arxiv	$78.53_{\pm 1.73}$	$71.25_{\pm 1.63}$	$80.25_{\pm 1.81}$	$73.18_{\pm 1.77}$	$80.68_{\pm 1.62}$	$73.71_{\pm 1.45}$	$76.30_{\pm 1.70}$	$70.00_{\pm 1.51}$	$76.61_{\pm 1.86}$	$70.68_{\pm 1.79}$	$77.53_{\pm 1.58}$	$70.67_{\pm 1.92}$
	MAG	$80.49_{\pm 1.59}$	$72.33_{\pm 1.84}$	$82.71_{\pm 1.63}$	$74.38_{\pm 1.70}$	$83.52_{\pm 1.79}$	$76.03_{\pm 1.62}$	$79.61_{\pm 1.88}$	$72.17_{\pm 1.85}$	$80.06_{\pm 1.78}$	$72.41_{\pm 1.66}$	$81.27_{\pm 1.82}$	$74.04_{\pm 1.61}$
6-MLP	Ali-CVR	$85.52_{\pm 1.66}$	$78.85_{\pm 1.92}$	$84.33_{\pm 1.80}$	$78.72_{\pm 1.65}$	$83.25_{\pm 1.93}$	$77.34_{\pm 1.84}$	$85.01_{\pm 1.75}$	$78.34_{\pm 1.78}$	$86.78_{\pm 1.57}$	$79.05_{\pm 1.91}$	$83.44_{\pm 1.63}$	$77.53_{\pm 1.72}$
	Arxiv	$80.81_{\pm 1.77}$	$73.77_{\pm 1.89}$	$79.23_{\pm 1.83}$	$72.52_{\pm 1.72}$	$78.11_{\pm 1.70}$	$71.54_{\pm 1.66}$	$80.55_{\pm 1.86}$	$73.44_{\pm 1.91}$	$81.03_{\pm 1.65}$	$74.00_{\pm 1.78}$	$78.63_{\pm 1.74}$	$71.61_{\pm 1.91}$
	MAG	$82.22_{\pm 1.91}$	$75.71_{\pm 1.76}$	$80.50_{\pm 1.74}$	$73.62_{\pm 1.91}$	$79.91_{\pm 1.82}$	$72.77_{\pm 1.85}$	$81.37_{\pm 1.66}$	$74.61_{\pm 1.73}$	$82.74_{\pm 1.59}$	$75.37_{\pm 1.90}$	$79.39_{\pm 1.83}$	$73.03_{\pm 1.88}$

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

attributes and edge ratio at 50%, with the rest vertically partitioned (Pang et al. 2022). As shown in Table 5, task accuracy remains stable with more parties due to embedding aggregation, while SGAC shows a slight drop as benign inputs dilute its influence. Nonetheless, SGAC maintains ASR and fidelity above 75.18% and 71.34%, even with 32 parties.

4.4 Adaptation Analysis and Ablation Study

We analyze SGAC's adaptation in surrogate generation and component utility with experiments on the Ali-CVR dataset.

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

Ablation Study. We evaluate *GEL*, *DKT*, and *SIC* via SGAC ablations (Table 6). Also, removing node- or edge-level saliency masking drops ASR by $11.67_{\pm 1.05}$ and $12.15_{\pm 0.67}$, while random edge dropping causes a $10.46_{\pm 1.52}$ decline,

Performance					Attribute/Edge Partition		
Dataset	#pa.	ASR	Fidelity	ori. acc.	Active	Passive	Attacker
Ali-CVR	2	$82.57_{\pm 1.35}$	$75.84_{\pm 1.47}$	$85.11_{\pm 0.68}$	50%	16.6%	16.6%
	4	$82.33_{\pm 1.42}$	$75.21_{\pm 1.53}$	$85.05_{\pm 0.45}$	50%	10%	10%
	8	$80.70_{\pm 1.47}$	$74.45_{\pm 1.66}$	$84.35_{\pm 0.35}$	50%	5.55%	5.55%
	16	$77.62_{\pm 1.35}$	$72.45_{\pm 1.80}$	$85.17_{\pm 0.89}$	50%	2.94%	2.94%
	32	$75.27_{\pm 1.20}$	$71.34_{\pm 1.34}$	$84.88_{\pm 1.05}$	50%	1.51%	1.51%

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

GEL	DKT	SIC	ASR	Fidelity
✓	✓	✓	85.55 ± 0.83	78.63 ± 0.61
✗	✗	✗	40.67 ± 1.14	30.48 ± 1.50
✓	✗	✗	53.50 ± 1.35	46.67 ± 0.78
✓	✓	✗	72.34 ± 0.88	78.63 ± 0.61

Table 6: Performance of SGAC and its variants.

and same-class additions bring only marginal gains of $0.64_{\pm 0.11}$ (Zhao et al. 2021; Xu et al. 2025).

4.5 Results against Countermeasures

We evaluate two types of countermeasures, i.e, gradient obfuscation (Zhu, Liu, and Han 2019) and adversarial training (Feng et al. 2021) against SGAC on the Ali-CVR dataset. ***(i) Noisy Gradients:*** Adding calibrated Laplace noise to the gradients before sharing. Figure 3(a) reveals that low levels of noise in gradients have minimal effect in mitigating SGAC. However, adding sufficient noise scale may flip the signs of gradients and counter SGAC. ***(ii) Gradient Filtering:*** Subsampling and retaining only the top θ_u fraction of noisy gradient entries each round to block low-magnitude signals. Figure 3(b) shows that $\theta_u \leq 0.3$ counters SGAC. However, this strict filtering also degrades clean classification accuracy to 60.47%. ***(ii) Adversarial Training:*** Incorporating simulated or SGAC-style adversarial inputs into the $\mathcal{M}_s$'s training loop. Figure 3(c) shows that such training can reduce ASR by up to 24.35% for the seen variant, but the unseen variant still achieves over 81.17% ASR, with both causing an average clean accuracy degradation of 18.65%.

5 Related Work

Conventional Adversarial Attacks. A common class of attacks assumes either full access to the target model (i.e., white-box attacks) or query access to predictions (i.e., black-box attacks) (Wu et al. 2019; Chang et al. 2023), directly utilizing real gradient for crafting. However, such access is often infeasible due to domain-specific knowledge and costly or privacy-sensitive queries in practice. An alternative line of work crafts transferable adversarial inputs with pre-trained surrogate models (i.e., gray-box attacks), such as SRLIM, SGAttack, and HGAttack (Liu et al. 2022; Li et al. 2023b; Zhao et al. 2024), to generate perturbations that generalize to the target model. However, they often require access to public model libraries, which may be unavailable in real-world tasks (Sun et al. 2020). Despite these advances, prior attacks manipulate full attributes and adjacency matrices, overlooking the GVFL setting in which an attacker controls only a subset of training/test data, and produces only partial input embeddings for the server model.

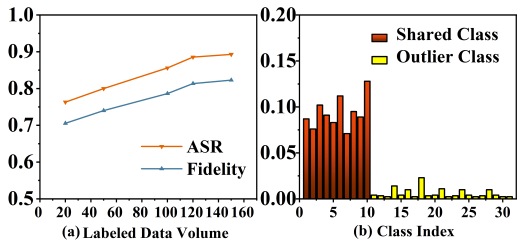

Figure 2: (a) Impact of labeled data volume. (b) Estimated class transferable probability $\mathbf{w}$.

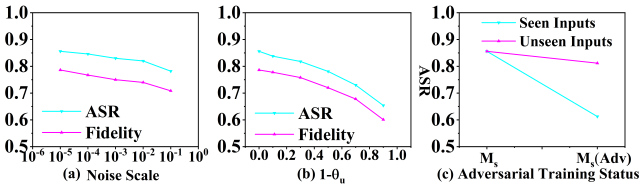

Figure 3: Defense performance of three methods: (a) noisy gradients, (b) gradient filtering, and (c) adversarial training.

Adversarial Attacks in GVFL. ADI was proposed to perturb attacker-controlled inputs to explicitly diminish benign parties' contributions and dominate the joint prediction (Pang et al. 2022). However, it relies on querying the server model and fetching test input features from benign parties. Graph-Fraudster (Chen et al. 2023) forges benign-party embeddings to craft adversarial inputs, but it requires access to the server model's architecture, gradients, and predictions. NA^2 (Chen, Zhang, and Zheng 2024) and Hijack (Qiu et al. 2022) build a surrogate model to approximate gradients and produce optimal perturbations on attacker's embeddings. Hijack additionally injects a poisoning process during training to amplify adversarial embedding's influence on prediction. However, they rely on training-domain labeled data for surrogate generation, which is difficult to acquire due to privacy or transmission reasons. In this paper, we propose SGAC, which operates exclusively during the inference and requires only auxiliary graphs from non-training domains.

6 Conclusion

In this paper, we proposed SGAC, the first adversarial attack framework against GVFL under a more relaxed setting. Unlike prior attacks, SGAC does not rely on knowledge of the server model and auxiliary data from the training domain. SGAC generates a high-fidelity surrogate 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 GVFL parties contributing important features. Extensive experiments demonstrate that SGAC achieves notable performance across various settings and significantly outperforms state-of-the-art attacks.

Acknowledgments

This work was supported by the Smart Grid-National Science and Technology Major Project of the Ministry of Science and Technology of China (Grant No. 2025ZD0808504).

References

Bai, Y.; Chen, Y.; Zhang, H.; Xu, W.; Weng, H.; and Goodman, D. 2023. {VILLAIN}: Backdoor Attacks Against Vertical Split Learning. In *USENIX Security 23*, 2743–2760.

Cao, Z.; Ma, L.; Long, M.; and Wang, J. 2018. Partial adversarial domain adaptation. In *ECCV 2018*, 135–150.

Chang, H.; Rong, Y.; Xu, T.; Huang, W.; Zhang, H.; Cui, P.; Wang, X.; Zhu, W.; and Huang, J. 2023. Adversarial Attack Framework on Graph Embedding Models With Limited Knowledge. *IEEE Trans. Knowl. Data Eng.*, 35(5): 4499–4513.

Chen, C.; Zhou, J.; Zheng, L.; Wu, H.; Lyu, L.; Wu, J.; Wu, B.; Liu, Z.; Wang, L.; and Zheng, X. 2022. Vertically Federated Graph Neural Network for Privacy-Preserving Node Classification. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 1959–1965. ijcai.org.

Chen, H.; Laine, K.; and Rindal, P. 2017. Fast private set intersection from homomorphic encryption. In *CCS 2017*, 1243–1255.

Chen, J.; Huang, G.; Zheng, H.; Yu, S.; Jiang, W.; and Cui, C. 2023. Graph-Fraudster: Adversarial Attacks on Graph Neural Network-Based Vertical Federated Learning. *IEEE Trans. Comput. Soc. Syst.*, 10(2): 492–506.

Chen, J.; Zhang, X.; and Zheng, H. 2024. Query-Efficient Adversarial Attack Against Vertical Federated Graph Learning. In *Attacks, Defenses and Testing for Deep Learning*, 75–96. Springer.

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

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting Adversarial Attacks with Momentum. In *CVPR 2018*, 9185–9193.

Feng, F.; He, X.; Tang, J.; and Chua, T. 2021. Graph Adversarial Training: Dynamically Regularizing Based on Graph Structure. *IEEE Trans. Knowl. Data Eng.*, 33(6): 2493–2504.

Fu, C.; Zhang, X.; Ji, S.; Chen, J.; Wu, J.; Guo, S.; Zhou, J.; Liu, A. X.; and Wang, T. 2022a. Label Inference Attacks Against Vertical Federated Learning. In *USENIX Security 2022*, 1397–1414.

Fu, X.; Zhang, B.; Dong, Y.; Chen, C.; and Li, J. 2022b. Federated Graph Machine Learning: A Survey of Concepts, Techniques, and Applications. *SIGKDD Explor.*, 24(2): 32–47.

Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; March, M.; and Lempitsky, V. 2016. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59): 1–35.

Guo, G.; Wang, C.; Yan, B.; Lou, Y.; Feng, H.; Zhu, J.; Chen, J.; He, F.; and Yu, P. S. 2023. Learning Adaptive Node Embeddings Across Graphs. *IEEE Trans. Knowl. Data Eng.*, 35(6): 6028–6042.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 1024–1034.

Hard, A.; Rao, K.; Mathews, R.; Beaufays, F.; Augenstein, S.; Eichner, H.; Kiddon, C.; and Ramage, D. 2018. Federated Learning for Mobile Keyboard Prediction. *CoRR*, abs/1811.03604.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Li, A.; Peng, H.; Zhang, L.; Huang, J.; Guo, Q.; Yu, H.; and Liu, Y. 2023a. FedSDG-FS: Efficient and Secure Feature Selection for Vertical Federated Learning. In *INFOCOM 2023*, 1–10.

Li, J.; Xie, T.; Chen, L.; Xie, F.; He, X.; and Zheng, Z. 2023b. Adversarial Attack on Large Scale Graph. *IEEE Trans. Knowl. Data Eng.*, 35(1): 82–95.

Liu, M.; Zhang, Z.; Tang, J.; Bu, J.; He, B.; and Zhou, S. 2024a. Revisiting, Benchmarking and Understanding Unsupervised Graph Domain Adaptation. *arXiv preprint arXiv:2407.11052*.

Liu, S.; Zou, D.; Zhao, H.; and Li, P. 2024b. Pairwise alignment improves graph domain adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, 32552–32575.

Liu, Z.; Luo, Y.; Zang, Z.; and Li, S. Z. 2022. Surrogate Representation Learning with Isometric Mapping for Gray-box Graph Adversarial Attacks. In Candan, K. S.; Liu, H.; Akoglu, L.; Dong, X. L.; and Tang, J., eds., *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*, 591–598. ACM.

Long, M.; Cao, Z.; Wang, J.; and Jordan, M. I. 2018. Conditional Adversarial Domain Adaptation. In *NeurIPS 2018*, 1647–1657.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS 2017*, volume 54 of *Proceedings of Machine Learning Research*, 1273–1282.

Ni, X.; Xu, X.; Lyu, L.; Meng, C.; and Wang, W. 2021. A vertical federated learning framework for graph convolutional network. *arXiv preprint arXiv:2106.11593*.

Pang, Q.; Yuan, Y.; Wang, S.; and Zheng, W. 2022. ADI: Adversarial Dominating Inputs in Vertical Federated Learning Systems. *arXiv preprint arXiv:2201.02775*.

Qiu, P.; Zhang, X.; Ji, S.; Li, C.; Pu, Y.; Yang, X.; and Wang, T. 2022. Hijack Vertical Federated Learning Models with Adversarial Embedding. *CoRR*, abs/2212.00322.

Saito, K.; Ushiku, Y.; and Harada, T. 2017. Asymmetric tri-training for unsupervised domain adaptation. In *ICML 2017*, 2988–2997.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *ICCV 2017*, 618–626.

Sim, R. H. L.; Zhang, Y.; Chan, M. C.; and Low, B. K. H. 2020. Collaborative machine learning with incentive-aware model rewards. In *ICML 2020*, 8927–8936.

Sun, Y.; Wang, S.; Tang, X.; Hsieh, T.-Y.; and Honavar, V. 2020. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of the Web Conference 2020*, 673–683.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2017. Graph Attention Networks. *CoRR*, abs/1710.10903.

Wang, K.; Shen, Z.; Huang, C.; Wu, C.-H.; Dong, Y.; and Kanakia, A. 2020. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1): 396–413.

Wang, S.; Gai, K.; Yu, J.; and Zhu, L. 2023. VFedMH: Vertical Federated Learning for Training Multi-party Heterogeneous Models. *CoRR*, abs/2310.13367.

Wei, C.; Wang, Y.; Bai, B.; Ni, K.; Brady, D.; and Fang, L. 2023. Boosting graph contrastive learning via graph contrastive saliency. In *International conference on machine learning*, 36839–36855. PMLR.

Wu, H.; Wang, C.; Tyshetskiy, Y.; Docherty, A.; Lu, K.; and Zhu, L. 2019. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv preprint arXiv:1903.01610*.

Xie, C.; Zhang, Z.; Wang, J.; Zhou, Y.; Ren, Z.; and Yuille, A. L. 2018. Improving Transferability of Adversarial Examples with Input Diversity. *CoRR*, abs/1803.06978.

Xie, Q.; Luong, M.; Hovy, E. H.; and Le, Q. V. 2020. Self-Training With Noisy Student Improves ImageNet Classification. In *CVPR 2020*, 10684–10695.

Xu, Y.; Huang, S.; Zhang, H.; and Li, X. 2025. Why does dropping edges usually outperform adding edges in graph contrastive learning? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 21824–21832.

Yao, D.; Li, S.; Xue, Y.; and Liu, J. 2024. Constructing Adversarial Examples for Vertical Federated Learning: Optimal Client Corruption through Multi-Armed Bandit. *arXiv preprint arXiv:2408.04310*.

Zhao, H.; Zeng, Z.; Wang, Y.; Ye, D.; and Miao, C. 2024. HGAttack: Transferable Heterogeneous Graph Adversarial Attack. In *IEEE International Conference on Agents, ICA 2024, Wollongong, Australia, December 4-6, 2024*, 100–105. IEEE.

Zhao, T.; Liu, Y.; Neves, L.; Woodford, O.; Jiang, M.; and Shah, N. 2021. Data augmentation for graph neural networks. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, 11015–11023.

Zhou, M.; Wu, J.; Liu, Y.; Liu, S.; and Zhu, C. 2020. Dast: Data-free substitute training for adversarial attacks. In *CVPR 2020*, 234–243.

Zhu, L.; Liu, Z.; and Han, S. 2019. Deep Leakage from Gradients. In *NeurIPS 2019*, 14747–14756.