HIDING IN PLAIN SIGHT: RELIABLE BACKDOOR ATTACK AGAINST VERTICAL FEDERATED GRAPH LEARNING VIA RECONSTRUCTION CONSISTENCY

Anonymous authors
Paper under double-blind review

ABSTRACT

Vertical federated graph neural networks (VFGNNs) enable collaborative learning from distributed graph data, but remain vulnerable to backdoor attacks. Unlike Euclidean data, graph data contain interconnected nodes, and message propagation may dilute trigger features, making backdoor attack methods developed for conventional vertical federated learning less effective when applied directly. In this paper, we propose Graph Camouflage (GraphCamo), a backdoor attack that combines structure-guided trigger learning and masked reconstruction-based backdoor consolidation. Its core idea is to select trigger support nodes based on graph topology and use masked feature reconstruction to preserve relevant information during graph aggregation. Experiments on four datasets and three GNN models show that, with only four anchor nodes, GraphCamo achieves an average attack success rate of 99.24% and an average standard deviation across configurations of 0.75%, while maintaining competitive main-task accuracy. Further experiments demonstrate its resistance to different defense mechanisms, while analyses of embeddings and gradients indicate that GraphCamo is less detectable.

1 INTRODUCTION

In recent years, graph neural networks (GNNs) have been widely applied to financial risk management, molecular property prediction, and traffic optimization for their advantages in modeling node attributes and edges (He et al., 2026; Nguyen et al., 2025; Yang et al., 2024). In practice, the same entity often uses services provided by different institutions, so its relevant features are distributed across these institutions (Liu et al., 2024; Mai & Pang, 2023). Because these institutions provide different services, they typically hold complementary information that would ideally be collected for joint modeling. However, privacy requirements and data barriers among these institutions limit the direct sharing of raw data (Wang et al., 2025; Zaeem & Barber, 2020). Vertical federated graph neural networks (VFGNNs) provide a way for different participants (clients) to perform model training without sharing their raw graph data (Chen et al., 2022a; Liu et al., 2024).

Although privacy concerns among different participants are alleviated, the distributed architecture of VFGNNs faces security threats, such as backdoor attacks (Yang et al., 2025). A malicious party can manipulate its local data and model so that the global model maintains its normal decision on clean inputs but outputs a target label when a trigger is present (Shen et al., 2025; Bai et al., 2023).

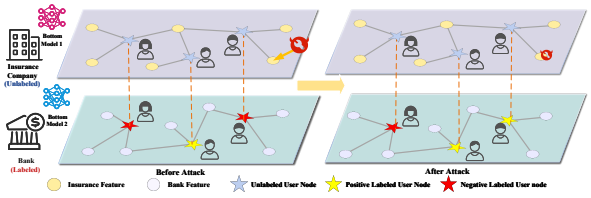


Figure 1: An example of backdoor attacks on VFGNNs.

Figure 1 illustrates an insurance company and a bank jointly conducting risk assessment: the bank holds customers' transaction information and risk labels, while the insurance company provides property-related features. If the insurance company is controlled by an attacker, it can implant a trigger pattern in its local data, causing the joint model to classify a fraudulent customer as a legitimate one. Because the active party cannot

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

directly inspect other participants' raw graph data or local models, such attacks could be difficult to detect and may cause serious losses if successfully carried out (Wang et al., 2026).

Many excellent studies have investigated backdoor attacks against vertical federated learning on traditional Euclidean data (Naseri et al., 2024; Gu & Bai, 2023; Chen et al., 2023; Bai et al., 2023). For example, VILLAIN uses label inference to guide poisoning (Bai et al., 2023), while LR-BA uses labeled auxiliary data to construct malicious latent representations (Gu & Bai, 2023). These studies typically adopt a strategy that uses label inference or prior knowledge to control a set of target samples, which aims to strengthen the association between the trigger and the target label. However, neighborhood aggregation in graph data makes node representations interdependent, and trigger features may be diluted during feature propagation. BVG expands the impact of poisoning by injecting triggers into several target nodes and their associated neighbors, which demonstrates the feasibility of backdoor attacks in VFGNNs (Yang et al., 2025). Yet these neighboring nodes do not necessarily belong to the same class as the target-class anchor nodes, and untargeted neighborhood injection may cause large fluctuations in attack success rate (ASR). Moreover, the randomness of injecting triggers into neighboring nodes may interfere with the normal feature representations of nodes from other classes and reduce main-task accuracy, a change that can also provide defenders with an important signal for detecting the attack (Sui et al., 2026; Bagdasaryan & Shmatikov, 2021; Lu et al., 2022). Therefore, to the best of our knowledge, no backdoor attack method has yet achieved both reliable attack performance and high main-task accuracy (MTA).

In this work, we propose Graph Camouflage (GraphCamo), a novel backdoor attack method for vertical federated graph learning that achieves high attack reliability (i.e., high ASR with low variability) under the precondition of maintaining high MTA. It comprises two modules: structure-guided trigger learning and masked reconstruction-based backdoor consolidation. The former selects trigger support nodes based on the malicious party's local graph structure, providing selective propagation support for trigger injection. The latter uses decoupled masked feature reconstruction to drive the shared encoder to exploit graph context while preserving both clean and trigger node information. Specifically, structure-guided trigger learning uses personalized PageRank scores computed from the local graph to select neighbors at different hop distances and then learns shared feature perturbations. Masked reconstruction-based backdoor consolidation uses the node representations from clean and triggered inputs to reconstruct their corresponding local features and employs one-sided gradient conflict projection to reduce interference from auxiliary learning with the main training process. The entire attack requires no changes to training labels, and trigger perturbations are injected only into the malicious party's local graph. Extensive experimental results demonstrate that the proposed approach consistently achieves an average ASR of 99.24% with minimal variation. More importantly, this attack performance is achieved without compromising the model's performance on clean samples, which demonstrates the practical feasibility of the proposed approach.

The main contributions are summarized as follows:

- We propose GraphCamo, an elegant backdoor attack against VFGNNs that achieves high ASR with low variability while preserving MTA. To explain the challenge, a conditional theoretical explanation of how graph propagation may attenuate trigger effects is provided.
- GraphCamo comprises two key modules. The trigger learning module uses graph topology to select support nodes, and the backdoor consolidation module uses reconstruction branches to recover clean and triggered node features from masked graph inputs.
- Across four datasets and three GNN models, GraphCamo achieves reliable attack performance, with an average ASR close to 100% while maintaining MTA competitive with seven attack baselines. GraphCamo also maintains high ASR under four representative defenses. Analyses of embeddings and gradients also suggest that the attack is difficult to detect.

2 Related Works

Backdoor Attacks Against VFL. In vertical federated learning (VFL), a malicious party controls only its local features or embeddings, without direct access to complete embeddings or ground-truth labels. Such backdoor attacks generally rely on auxiliary samples or prior knowledge for label inference to guide trigger construction and backdoor injection. VILLAIN starts with a known target-class sample and identifies additional samples through changes in the norms of gradients returned

before and after embedding swapping (Bai et al., 2023). BadVFL (Naseri et al., 2024) uses labeled auxiliary data from the same feature distribution and label space to infer labels and guide trigger injection. Chen et al. (2024) proposed UAB, which selects pseudo-labeled samples by ranking gradient norms but relies on class skew in binary classification. LR-BA (Gu & Bai, 2023) trains a label-inference head on a small labeled auxiliary set, then fine-tunes the local model to produce malicious latent representations for triggered inputs. These studies reveal backdoor risks in VFL. However, connections between graph nodes may dilute backdoor features through neighborhood aggregation, which may weaken the mapping between the trigger and the target label.

Security Threats to VFGNNs. Vertical federated graph neural networks (VFGNNs) extend VFL to graph data but remain vulnerable to various security threats. Chen et al. (2022b) proposed GraphFraudster, which constructs a shadow model of the server using leaked global node embeddings and generates adversarial perturbations to manipulate joint predictions. Regarding privacy leakage, Zari et al. (2024) identified shared gradients as a source of link information leakage and demonstrated that they can reveal private edges held by other parties. SGAC (Liu et al., 2026) adopts auxiliary graphs to construct a surrogate model and generate transferable adversarial inputs for manipulating joint inference. Regarding backdoor attacks, BVG (Yang et al., 2025) combines multi-hop triggers with a backdoor retention mechanism to induce targeted misclassification using only a few known target-class nodes. However, neighboring nodes may belong to non-target classes, and perturbing them could alter their feature representations, potentially degrading main-task accuracy.

3 PRELIMINARIES

Problem Formulation. Consider a VFGNN with N parties with node set V. Party i holds a local graph $G_i = (V, E_i, X_i)$ with private edges E_i and node features X_i. The complete feature matrix and edge set are $X = X_1 \| \cdots \| X_N$ and $E = \bigcup_{i=1}^{N} E_i$. Predefined indices partition V into training, validation, and test sets V_{tr}, V_{val}, and V_{te}. The active party holds node labels. Each party's bottom GNN f_i, parameterized by θ_i, produces $z_{i,v} = f_i(G_i; \theta_i)_v$ for node v.

The active party passes $z_{1,v}, \ldots, z_{N,v}$ through the top model f_g and a softmax layer to obtain class probabilities p_v and prediction $\widehat{y}_v = \arg\max_c p_{v,c}$. With model parameters $\Theta = \{\theta_1, \ldots, \theta_N, \theta_{\mathrm{top}}\}$, the classification objective for a batch $\mathcal{B} \subseteq V_{\mathrm{tr}}$ is

$$\mathcal{L}_{\mathrm{cls}}(\Theta) = -\frac{1}{|\mathcal{B}|} \sum_{v \in \mathcal{B}} \log p_{v,y_v}. \tag{1}$$

After updating θ_{top}, the active party returns $q_{i,v} = \partial \mathcal{L}_{\mathrm{cls}} / \partial z_{i,v}$ to party i. With $q_{i,v}$ fixed, party i backpropagates through $\mathcal{L}_i^{\mathrm{bp}} = \sum_{v \in V} z_{i,v}^{\mathsf{T}} q_{i,v}$. Since $q_{i,v} = 0$ outside $\mathcal{B}$, the sum is effectively over the batch, and its θ_i gradient equals the chain-rule gradient of $\mathcal{L}_{\mathrm{cls}}$ through f_i.

Consider one malicious passive party m with target class y_t and a small anchor node set $\mathcal{A} \subseteq \{v \in V_{\mathrm{tr}} : y_v = y_t\}$. Without changing node labels, it constructs a perturbation matrix Δ over its local features. The trigger transformation and resulting representation are

$$\mathcal{T}_\Delta(G_m) = (V, E_m, X_m + \Delta), \qquad \widetilde{z}_{m,v} = f_m(\mathcal{T}_\Delta(G_m); \theta_m)_v. \tag{2}$$

For an attack set $V_{\mathrm{atk}} \subseteq \{v \in V_{\mathrm{te}} : y_v \neq y_t\}$, the desired top-model output is

$$\begin{cases} \arg\max_c f_g(z_{1,v}, \ldots, \widetilde{z}_{m,v}, \ldots, z_{N,v})_c = y_t, & v \in V_{\mathrm{atk}}, \\ \arg\max_c f_g(z_{1,v}, \ldots, z_{m,v}, \ldots, z_{N,v})_c = y_v, & v \in V_{\mathrm{te}} \setminus V_{\mathrm{atk}}. \end{cases} \tag{3}$$

Adversary Capabilities. The adversary controls passive party m and follows the prescribed communication protocol, including timely uploads of local embeddings. It can access its local data, local model, and the gradients returned by the active party, but it neither controls the top model nor participates in label management or loss computation at the active party. Consistent with prior VFL backdoor settings (Gu & Bai, 2023; Naseri et al., 2024; Yang et al., 2025), the adversary knows a few target-class samples, which can be purchased or obtained through social engineering.

Attack Objective. When the adversary injects the trigger into a test node, the top model classifies the node as the target class y_t. Meanwhile, the top model should preserve its predictive performance

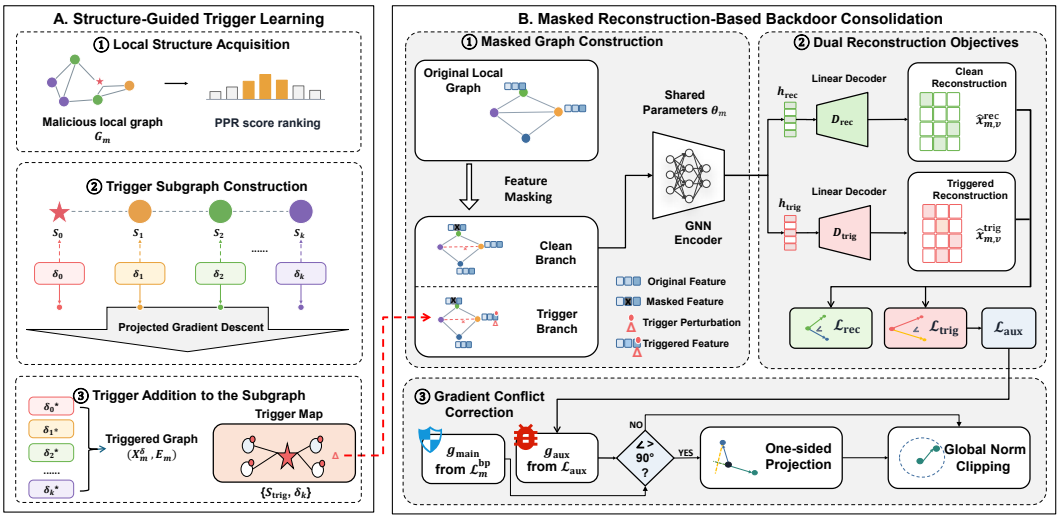


Figure 2: Overview of GraphCamo.

on nodes without the trigger. A targeted backdoor attack can be viewed as a more constrained form of an untargeted attack: an untargeted attack only requires any incorrect prediction, whereas a targeted attack must consistently induce a prespecified class and is therefore more challenging.

4 Proposed Method

Figure 2 illustrates GraphCamo, and Algorithm 1 in Appendix A summarizes its procedure.

4.1 Structure-Guided Trigger Learning

Local Structure Acquisition. Given a maximum propagation order K, for each $a \in \mathcal{A}$ in the malicious party's local graph G_m, where $\mathcal{A}$ denotes the anchor nodes with the target label y_t, the nodes whose shortest-path distance from a is exactly k are denoted by $\mathcal{C}_{a,k} = \{v \in V : \mathrm{dist}_{G_m}(a, v) = k\}$ for $k \in \{0, \ldots, K\}$. Let $\mathcal{U}_a = \bigcup_{k=0}^{K} \mathcal{C}_{a,k}$ be the set of potential poisoning nodes within K hops of a. The associated subgraph G_a is formed by $\mathcal{U}_a$ and the local edges $E_a = \{(u, v) \in E_m : u \in \mathcal{U}_a,\ v \in \mathcal{U}_a\}$. The transition matrix P_a of this induced subgraph is defined as

$$(P_a)_{uv} = \begin{cases} \dfrac{1}{\deg_a(u)}, & v \in \mathcal{N}_a(u), \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where $\mathcal{N}_a(u)$ is the set of neighbors of u in G_a and $\deg_a(u) = |\mathcal{N}_a(u)|$ is its local degree. The transition probability of a node is distributed uniformly among its local neighbors. The row for a node without neighbors is set to zero. Initialized with the one-hot vector r_a of anchor node a, personalized PageRank (PPR) (Wang et al., 2020) is computed as

$$\pi_a^{(\ell+1)} = (1 - d) r_a + d P_a^{\mathsf{T}} \pi_a^{(\ell)}. \tag{5}$$

The score $(\pi_a)_v$ measures the tendency of node v to be visited by a random walk restarting at a.

Trigger Subgraph Construction. For hop k, all anchor nodes are first excluded from the candidate set: $\mathcal{C}_{a,k}^{\mathrm{cand}} = \mathcal{C}_{a,k} \setminus \mathcal{A}$, where $k \in \{1, \ldots, K\}$. Let $\rho \in (0, 1]$ denote the fraction of candidates to be selected. Each nonempty candidate set is ranked by PPR score in descending order, and $b_{a,k} = \max(1, \lceil \rho |\mathcal{C}_{a,k}^{\mathrm{cand}}| \rceil)$ nodes are selected from the middle of the ranking to form $\mathcal{M}_{a,k}$.

Selections are made independently for each anchor and merged into $\mathcal{U}_{\mathrm{sel}} = \bigcup_{a \in \mathcal{A}} \bigcup_{k=1}^{K} \mathcal{M}_{a,k}$. The core neighbor set at shortest distance k from the anchor node set is defined as $\mathcal{C}_k = \{v : \min_{a \in \mathcal{A}} \mathrm{dist}_{G_m}(a, v) = k\}$, yielding $\mathcal{S}_0 = \mathcal{A}$ and $\mathcal{S}_k = \mathcal{C}_k \cap \mathcal{U}_{\mathrm{sel}}$ for $k \in \{1, \ldots, K\}$. If a node is selected by multiple anchor nodes, it is retained only once in $\mathcal{U}_{\mathrm{sel}}$. Its shortest distance to the anchor set $\mathcal{A}$ determines its assignment, ensuring that no node is repeated in the trigger subgraph.

Trigger Addition to the Subgraph. Let $\mathcal{S}_{\text{trig}} = \bigcup_{k=0}^{K} \mathcal{S}_k$. On this support, δ_k denotes the trigger shared by nodes at hop k. Following prior work (Madry et al., 2018; Huang et al., 2021), it is optimized using the PGD method by

$$\Delta_v = \begin{cases} \delta_k, & v \in \mathcal{S}_k, \quad k \in \{0, \dots, K\}, \\ \mathbf{0}, & v \notin \mathcal{S}_{\text{trig}}. \end{cases} \tag{6}$$

Triggered representations are generated by the malicious party using $X_m^{\delta} = X_m + \Delta$ and the original local edges E_m. The classification loss is computed by the active party using the true labels, and representation gradients are returned. Using the local backpropagation $\mathcal{L}_m^{\text{bp}}$, the malicious party averages the perturbation gradients over each $\mathcal{S}_k$ and performs the update:

$$g_k = \frac{1}{|\mathcal{S}_k|} \sum_{v \in \mathcal{S}_k} \frac{\partial \mathcal{L}_m^{\text{bp}}}{\partial \Delta_v}, \qquad \delta_k \leftarrow \text{clip}(\delta_k - \alpha \, \text{sign}(g_k), -\epsilon, \epsilon) \, . \tag{7}$$

Here, α is the update step size, and ϵ is the perturbation bound. Equation 7 applies only when $\mathcal{S}_k \neq \emptyset$. Otherwise, gradient averaging and the trigger update are skipped, leaving δ_k unchanged.

4.2 Masked Reconstruction-Based Backdoor Consolidation

Masked Graph Construction. To construct the auxiliary masked graph, an element-wise retention mask is independently sampled for the original local features:

$$M_{v,j} \sim \text{Bernoulli}(1 - r_x), \qquad X_m^{\text{mask}} = M \odot X_m. \tag{8}$$

In addition, undirected edge pairs are dropped from E_m in a fixed manner to obtain E_m^{drop}. The same feature mask and dropped-edge result are used by both reconstruction branches. The clean branch directly encodes X_m^{mask}. In the triggered branch, the newly updated Δ is added after masking.

Dual Reconstruction Objectives. Two linear decoders with no shared parameters, D_{rec} and D_{trig}, are used to map the representations of the two inputs back to the local feature space:

$$\widehat{x}_{m,v}^{\text{rec}} = D_{\text{rec}}\big(f_m(X_m^{\text{mask}}, E_m^{\text{drop}}; \theta_m)_v\big) \, , \qquad \widehat{x}_{m,v}^{\text{trig}} = D_{\text{trig}}\big(f_m(X_m^{\text{mask}} + \Delta, E_m^{\text{drop}}; \theta_m)_v\big) \, . \tag{9}$$

The clean target is $x_{m,v}$, and the triggered target is $(X_m + \Delta)_v$. Let $\ell_{\cos}(u, v) = [\max(1 - c(u, v), 0)]^2$, where

$$c(u, v) = \frac{u^{\mathsf{T}} v}{\max(\|u\|_2, \varepsilon_{\text{num}}) \max(\|v\|_2, \varepsilon_{\text{num}})}.$$

The two losses are

$$\mathcal{L}_{\text{rec}} = \frac{1}{|\mathcal{B}|} \sum_{v \in \mathcal{B}} \ell_{\cos}(\widehat{x}_{m,v}^{\text{rec}}, x_{m,v}), \qquad \mathcal{L}_{\text{trig}} = \frac{1}{|\mathcal{S}_{\text{trig}}|} \sum_{v \in \mathcal{S}_{\text{trig}}} \ell_{\cos}(\widehat{x}_{m,v}^{\text{trig}}, (X_m + \Delta)_v). \tag{10}$$

The clean branch constrains the recoverability of original features for nodes in the current training batch. The triggered branch extends direct reconstruction supervision to all nodes in the trigger support. The auxiliary objective is $\mathcal{L}_{\text{aux}} = \lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \lambda_{\text{trig}} \mathcal{L}_{\text{trig}}$. For the auxiliary task, Δ is fixed, and only the malicious encoder and the decoders are updated. This constraint is intended to preserve information from both clean and triggered features.

Gradient Conflict Correction. The reconstruction objective may produce an update direction that conflicts with the main training objective. Let $g_{\text{main}} = \nabla_{\theta_m} \mathcal{L}_m^{\text{bp}}$ denote the gradient with respect to the malicious party's parameters retained from the main step, and let $g_{\text{aux}} = \nabla_{\theta_m} \mathcal{L}_{\text{aux}}$ be the auxiliary gradient obtained by a new forward pass after the main update. A global inner product is computed over all valid encoder parameters. If it is negative, only the auxiliary gradient is modified:

$$g_{\text{aux}}^{\perp} = \begin{cases} g_{\text{aux}} - \dfrac{\langle g_{\text{aux}}, g_{\text{main}} \rangle}{\|g_{\text{main}}\|_2^2} g_{\text{main}}, & \langle g_{\text{aux}}, g_{\text{main}} \rangle < 0, \\ g_{\text{aux}}, & \text{otherwise.} \end{cases} \tag{11}$$

After global norm clipping, the gradient updates the malicious encoder a second time:

$$g_{\text{aux}}^{\text{use}} = g_{\text{aux}}^{\perp} \min\left(1, \frac{C}{\|g_{\text{aux}}^{\perp}\|_2}\right). \tag{12}$$

Decoder gradients are updated independently, without projection or clipping.

5 THEORETICAL ANALYSIS

To isolate propagation effects, we compare the same feature-level perturbation under fixed linear bottom and top models. Node features $x_{m,v}$, representations $z_{m,v}$, and perturbations Δ_v are columns, stored as transposed rows in X_m, Z_m, and Δ. The malicious party's bottom model is

$$Z_m^{\mathrm{VFL}} = X_m W_m, \qquad Z_m^{\mathrm{graph}} = H X_m W_m, \qquad H = S_m^L, \tag{13}$$

Here, W_m is the fixed feature transformation, $L \geq 1$, and $S_m = \widetilde{D}_m^{-1/2}(A_m + I)\widetilde{D}_m^{-1/2}$ is the SGC propagation operator (Wu et al., 2019). The local graph is undirected and unweighted. A_m is its adjacency without self-loops, and $\widetilde{D}_m$ contains the degrees after adding them. The non-graph model uses $H = I$.

Proposition 1 (Trigger attenuation under graph propagation)**.** *For a perturbation Δ_v applied only to node v, the representation changes at that node satisfy*

$$\Delta z_{m,v}^{\mathrm{VFL}} = W_m^{\mathsf{T}} \Delta_v, \qquad \Delta z_{m,v}^{\mathrm{graph}} = H_{vv} W_m^{\mathsf{T}} \Delta_v, \quad 0 \leq H_{vv} \leq 1. \tag{14}$$

For one propagation layer, $H_{vv} = 1/(\deg_m(v) + 1)$, where $\deg_m(v)$ is the degree of v in the original local graph G_m. Let F contain the logits produced by the active party's linear top model f_g, and let B_m be its weight block for party m. Then $F = Z_m B_m + F_{-m}$, where F_{-m} collects the fixed contributions from all parties other than m, together with the top-model bias. For target class y_t and competing class c, define the classification margin $\Gamma_{v,c} = F_{v,y_t} - F_{v,c}$. Its increment is

$$\Delta\Gamma_{v,c}^{\mathrm{graph}} = H_{vv} \Delta\Gamma_{v,c}^{\mathrm{VFL}} = H_{vv} \Delta_v^{\mathsf{T}} W_m (B_{m,:,y_t} - B_{m,:,c}). \tag{15}$$

A trigger contribution sufficient to overcome a fixed negative margin without propagation may become insufficient after aggregation. Appendix B.1 gives the proof.

Fix one branch, its encoder and decoder D, Δ, and the dropped-edge graph. Sample a node uniformly from a fixed batch for clean reconstruction (CR) or $\mathcal{S}_{\mathrm{trig}}$ for triggered reconstruction (TR), and a fresh feature mask as in Equation 8. For its unmasked target x_{tar} and masked-input representation Z, define $T = \mathrm{nrm}(x_{\mathrm{tar}})$, $U = \mathrm{nrm}(D(Z))$, and $R = \mathbb{E}\ell_{\cos}(D(Z), x_{\mathrm{tar}})$, where $\mathrm{nrm}(x) = x/\|x\|_2$ for $x \neq 0$ and $\mathrm{nrm}(0) = 0$.

Proposition 2 (Recoverable feature directions)**.** *Suppose the target and decoder-output norms are at least $\varepsilon_{\mathrm{num}}$ almost surely. Then*

$$\mathbb{E}\|T - \mathbb{E}[T \mid Z]\|_2^2 \leq \mathbb{E}\|T - U\|_2^2 \leq 2\sqrt{R}. \tag{16}$$

Furthermore, if T and Z are independent, then

$$R \geq \left(1 - \|\mathbb{E}T\|_2\right)^2. \tag{17}$$

The first term in Equation 16 is the minimum mean squared error for recovering target directions from Z. Equation 17 gives a lower bound for target-independent representations. CR and TR require recovery of original and triggered feature directions, respectively, encouraging the shared encoder to preserve this information after graph aggregation. The proof is provided in Appendix B.2.

6 EXPERIMENTS

6.1 EXPERIMENTAL SETTINGS

Datasets. We conduct experiments on four datasets: Cora and Cora_ml (McCallum et al., 2000), CiteSeer (Sen et al., 2008), and PubMed (Namata et al., 2012). Appendix C.1 provides dataset statistics (Table 2). All parties hold the same node set. Node features are evenly divided between the two parties along the feature dimension, while the edge set is partitioned into two complementary local edge sets with no overlap. This is consistent with the experimental setting in Yang et al. (2025).

Models. To examine the applicability of the attack method to different graph message passing mechanisms, we use GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and SGC (Wu et al., 2019) as local bottom models. Details of these models are described in Appendix C.2.

Table 1: Performance (%) of different backdoor attack methods on four datasets. Higher ASR and MTA indicate stronger attack effectiveness and better main-task performance, respectively.

Local Model	Datasets	ASR							
		GF	TECB	VILLAIN	ET	NFT	NFR	BVG	GraphCamo
GCN	Cora	42.05±20.59	34.83±18.59	85.59±11.72	56.18±25.39	67.54±27.13	43.93±23.63	42.46±15.31	**98.02±2.41**
	Cora_ml	16.04±4.91	19.34±5.68	48.03±20.51	57.10±18.27	45.82±22.35	32.20±7.29	88.50±15.27	**99.95±0.09**
	CiteSeer	46.52±37.91	75.30±10.41	82.41±25.66	83.49±9.02	95.71±4.49	52.52±23.75	80.20±23.68	**99.63±0.24**
	PubMed	93.85±5.21	69.93±22.75	80.24±8.11	74.80±11.91	71.32±11.27	69.22±24.29	99.90±0.16	**100.00±0.00**
GAT	Cora	31.66±37.59	37.98±15.04	63.10±19.18	45.93±18.47	73.43±22.37	25.93±14.71	91.31±11.37	**98.23±1.36**
	Cora_ml	16.60±10.36	23.35±19.09	33.47±18.02	40.13±12.02	56.65±34.27	25.71±7.22	**100.00±0.00**	**100.00±0.00**
	CiteSeer	31.90±15.47	64.90±25.18	86.09±8.48	70.99±18.55	88.00±20.54	55.62±11.75	94.02±6.21	**96.81±2.36**
	PubMed	94.34±9.73	79.17±21.17	85.78±23.16	74.07±13.31	88.61±18.46	66.00±22.06	**100.00±0.00**	**100.00±0.00**
SGC	Cora	37.93±23.37	46.53±23.66	62.09±23.94	51.03±21.28	61.29±21.75	36.53±11.09	77.29±20.47	**98.25±2.54**
	Cora_ml	11.29±5.21	15.99±5.82	23.65±14.64	61.84±14.23	41.93±14.48	32.69±8.70	97.13±3.10	**99.99±0.02**
	CiteSeer	35.86±23.44	71.79±9.70	78.31±17.62	86.50±6.51	90.18±6.80	66.89±9.99	93.91±5.76	**100.00±0.00**
	PubMed	80.85±34.00	66.00±29.13	81.73±11.50	72.41±20.93	77.20±17.67	58.17±24.12	96.49±7.79	**100.00±0.00**
Local Model	**Datasets**	**MTA**							
		GF	TECB	VILLAIN	ET	NFT	NFR	BVG	GraphCamo
GCN	Cora	54.64±10.79	59.04±12.84	56.28±10.30	58.60±11.80	59.56±12.52	55.88±12.55	60.96±6.98	**66.06±2.72**
	Cora_ml	74.74±2.34	74.50±3.76	74.30±3.23	74.69±3.22	75.65±1.90	75.38±2.12	74.40±3.29	**77.51±1.34**
	CiteSeer	40.86±19.65	50.12±5.38	**53.64±1.86**	51.88±5.70	53.08±6.30	53.18±3.54	49.98±2.93	53.58±2.02
	PubMed	57.14±6.25	46.44±22.07	55.44±22.08	60.04±17.42	66.96±4.30	49.92±24.55	72.74±1.74	**74.18±1.08**
GAT	Cora	55.62±20.58	59.58±10.03	58.00±13.20	64.32±4.62	65.24±2.11	64.46±1.54	61.84±4.17	**65.72±1.60**
	Cora_ml	74.48±7.40	74.64±5.18	75.21±3.08	77.51±2.47	78.09±2.36	76.71±2.80	74.05±3.37	**78.36±2.29**
	CiteSeer	48.32±9.57	50.38±3.26	49.96±2.07	50.66±2.72	50.24±5.08	47.08±8.62	47.42±5.17	**51.76±2.67**
	PubMed	51.06±20.53	39.82±21.96	47.88±24.65	63.22±12.73	72.10±1.79	57.04±17.34	**74.06±1.59**	73.94±0.23
SGC	Cora	55.86±14.73	53.86±17.88	**67.96±1.44**	67.48±1.87	67.54±3.19	67.14±0.97	65.06±3.89	65.96±4.55
	Cora_ml	78.44±2.57	79.60±1.21	**79.80±1.56**	77.61±1.86	78.04±1.61	77.58±2.37	77.76±2.24	79.56±0.98
	CiteSeer	49.90±13.55	52.22±1.53	52.70±2.24	51.12±7.03	**56.90±6.07**	53.06±3.71	53.54±3.53	55.38±1.69
	PubMed	64.76±6.06	45.50±19.98	71.20±1.48	66.94±4.99	67.06±4.46	65.58±6.71	74.12±0.62	**74.86±0.34**

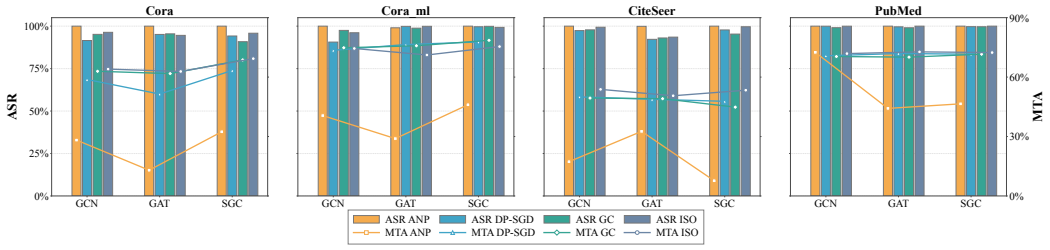


Figure 3: Results of GraphCamo under different defense mechanisms.

Attack Baselines. We compare GraphCamo with seven attack baselines: Graph-Fraudster (GF) (Chen et al., 2022b), TECB (Chen et al., 2023), VILLAIN (Bai et al., 2023), Edge Triggering (ET), Node Feature Triggering (NFT), Node Feature Replacement (NFR), and BVG (Yang et al., 2025). Their descriptions and attack configurations are provided in Appendix C.3.

Details and Metrics. The experiment involves two parties, including one malicious passive party. The model is trained for 30 epochs: the first five epochs are used for clean warm-up, and the following 25 epochs perform main-task training and backdoor injection. GraphCamo uses a two-hop trigger. For each anchor node, local Personalized PageRank (PPR) ranks the one-hop and two-hop candidate nodes separately, and the middle 50% of each ranking is retained as trigger-support nodes. We set class 0 as the attack target. Training details and hyperparameters are given in Appendix C.4.

We use attack success rate (ASR) and main-task accuracy (MTA) to evaluate attack effectiveness and clean-task performance, which are de facto standards in this field (Bai et al., 2023; Yang et al., 2025; Gu & Bai, 2023). We also assess backdoor stealthiness by the mean squared error (MSE) of uploaded embeddings and returned gradients. Definitions of these metrics are provided in Appendix C.5.

6.2 Attack Performance

Table 1 compares GraphCamo with the baselines on four datasets and three different models. Among the reported results, our method achieves the highest ASR across all experimental settings, reaching nearly 100% in most cases. Meanwhile, the average standard deviation of GraphCamo's ASR is only 0.75%, compared with 9.09% for BVG. These results indicate that GraphCamo achieves both

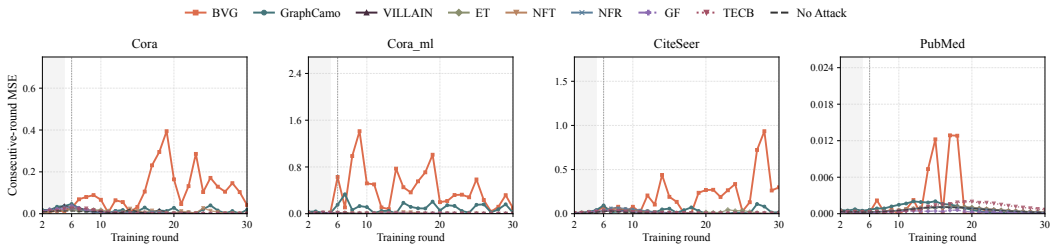


Figure 4: Embedding MSE with GCN as the bottom model.

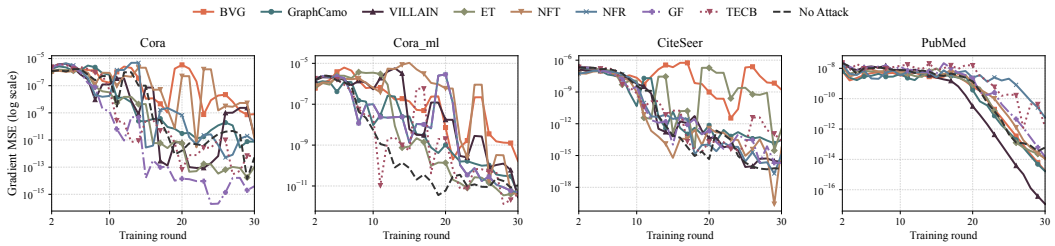


Figure 5: Gradient MSE with GCN as the bottom model.

higher attack success rates and more consistent attack performance. This may be because BVG does not prioritize influential nodes, which brings greater variability in attack performance.

GraphCamo achieves the highest MTA in most settings and trails the best by no more than 2% in the rest. In some settings, GraphCamo has a slightly lower MTA than VILLAIN. This may be because VILLAIN mainly injects trigger patterns at the intermediate representation level and therefore has a relatively limited effect on graph propagation. However, its slightly higher MTA does not correspond to reliable backdoor effectiveness. These results demonstrate that GraphCamo achieves reliable attack performance while maintaining high MTA.

6.3 Resistance to Defense

Evaluation under Defense Methods. To assess GraphCamo's robustness under defense mechanisms, we consider adversarial neuron pruning (ANP) (Wu & Wang, 2021), differentially private SGD (DP-SGD) (Abadi et al., 2016), gradient compression (GC) (Fu et al., 2022; Kairouz et al., 2021), and isotropic noise (ISO) (Li et al., 2022). The defense configurations are detailed in Appendix C.6. Figure 3 compares GraphCamo under these four defenses. GraphCamo maintains a high attack success rate under all four defenses, demonstrating strong resistance to these defenses. MTA remains generally high under DP-SGD, GC, and ISO. ANP produces lower MTA than the other three defenses in most configurations, while the corresponding ASR remains high. These results demonstrate the robustness of GraphCamo against defenses.

Backdoor Stealthiness. The active party can observe the malicious party's uploaded embeddings and the gradients it returns. We assess backdoor stealthiness by measuring the MSE of these signals between consecutive epochs. Figures 4 and 5 show the GCN results. GraphCamo's embedding MSE is generally low and smooth, whereas BVG exhibits pronounced spikes. GraphCamo's gradient MSE also tends to decline as training progresses. On PubMed, its gradient MSE follows a broadly similar trend to the no-attack baseline. These results show that the attack may be less conspicuous to the active party. The details and results for GAT and SGC are provided in Appendix D.1.

6.4 Ablation Studies and Further Analysis

Impact of Masked Feature Reconstruction. Figure 6 includes three ablation settings related to the reconstruction module. After removing trigger feature reconstruction (w/o TR), the overall ASR decreases, while MTA changes only slightly. This suggests that the branch helps preserve trigger information and supports learning the association between the trigger and the target class. Removing clean feature reconstruction (w/o CR) leads to notable drops in both ASR and MTA. This shows that the branch helps preserve useful representations of clean nodes. When both reconstruction

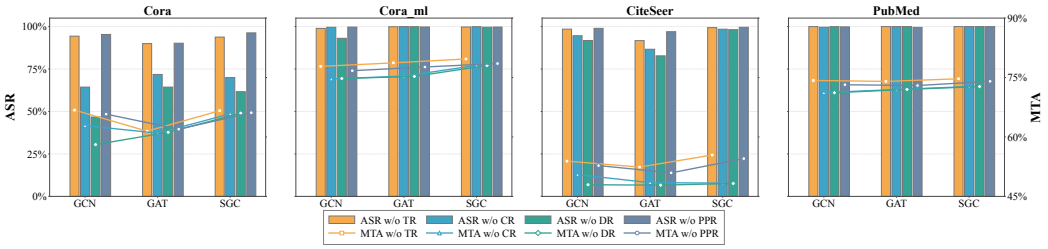


Figure 6: Ablation results. TR and CR denote trigger and clean feature reconstruction, respectively. DR denotes the combination of both reconstruction branches. PPR denotes personalized PageRank neighbor selection. "w/o" means that the corresponding component is removed.

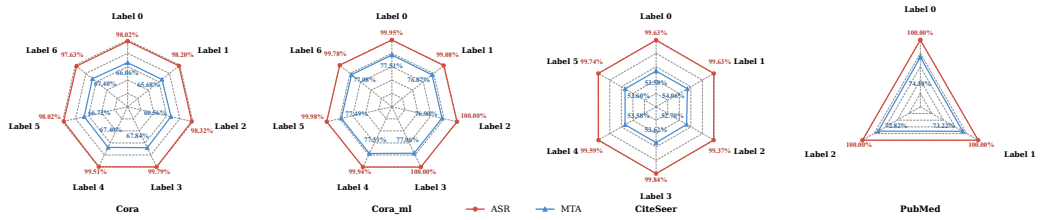


Figure 7: Results of GraphCamo with different target labels using GCN as the bottom model. Attack success rate (ASR) and main-task accuracy (MTA) are shown in red and blue, respectively.

branches are removed (w/o DR), performance declines further. This indicates that the two branches complement each other to improve attack reliability while preserving MTA.

Impact of PPR Neighbor Selection. After PPR neighbor selection is removed (w/o PPR), the overall attack success rate decreases, and this effect is more evident on some datasets such as Cora. PPR selects trigger-support nodes according to their local structural association with the anchor node, allowing trigger information to propagate along more suitable neighborhood paths. Therefore, PPR neighbor selection can reduce interfering injections and improve the attack effectiveness of GraphCamo under different graph structures and bottom models.

Impact of Number of Anchor Nodes. GraphCamo achieves high ASR with only 4 anchor nodes; increasing their number does not consistently improve attack performance. The full comparison across 4, 6, 8, and 10 anchor nodes is provided in Appendix D.2.

Impact of Number of Participants. As the number of participants increases, GraphCamo maintains ASR close to 100% on Cora_ml and PubMed, while ASR shows a certain degree of decline on Cora and CiteSeer. The complete results and further analysis are provided in Appendix D.3.

Impact of Target Label. To examine the effect of target label selection, GraphCamo is evaluated across all target labels in each dataset using different bottom models, as shown in Figure 7. Only the results with GCN across all datasets are presented here. Results with the other bottom models are provided in Appendix D.4. GraphCamo maintains generally high and stable ASR and MTA across different target labels, consistent with the main experimental results. These results indicate that GraphCamo's attack effectiveness is not limited to a particular target class.

7 Conclusion

In this paper, we have developed GraphCamo, a backdoor attack against VFGNNs designed to achieve reliable attack success while preserving main-task accuracy. GraphCamo selects trigger support nodes using local PPR and consolidates the backdoor through separate masked reconstruction of clean and triggered features. Extensive experiments show that GraphCamo achieves an average ASR approaching 100% with low variability while maintaining competitive MTA compared with seven attack baselines. GraphCamo also maintains high ASR under four representative defenses. Analyses of embeddings and gradients further indicate that the attack is not easily detectable. These findings highlight the security vulnerabilities of VFGNNs and the need for tailored defense methods.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing and improving clarity of expression. The authors take full responsibility for the final content of the paper.

ETHICS STATEMENT

This research aims to identify backdoor security risks in vertical federated graph learning and inform future detection and defense research. All experiments were conducted on publicly available datasets in controlled, simulated federated learning environments. We recognize the dual-use potential of the attack methods studied in this work and intend these research findings to support security evaluations and academic research. If reviewers or members of the research community raise ethical concerns about this work, we will address them transparently and responsibly.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 4 describes GraphCamo's trigger learning and dual-branch masked reconstruction, while Section 6.1 summarizes the datasets, models, baselines, training setup, and evaluation metrics. Appendix A gives the training algorithm, and Appendix B contains the theoretical proofs. Appendix C.1 provides dataset descriptions and statistics (Table 2). Appendix C.2 describes the local GNN models and the active party's representation aggregation. Appendix C.3 describes the attack baselines, their graph adaptations, and configurations. Appendix C.4 lists training details and hyperparameters, Appendix C.5 defines the evaluation metrics, and Appendix C.6 specifies the defense configurations. Appendices D.1, D.2, D.3, and D.4 provide further stealthiness, anchor-node, participant-count, and target-label results, respectively. To facilitate reproducibility, we submit anonymized source code as supplementary material. Together with the descriptions in the main text and appendices, the code will support reproduction of our experiments.

REFERENCES

Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, 2016. doi: 10.1145/2976749.2978318.

Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1505–1521, 2021.

Yijie Bai, Yanjiao Chen, Hanlei Zhang, Wenyuan Xu, Haiqin Weng, and Dou Goodman. VILLAIN: Backdoor attacks against vertical split learning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 2743–2760, 2023.

Chaochao Chen, Jun Zhou, Longfei Zheng, Huiwen Wu, Lingjuan Lyu, Jia Wu, Bingzhe Wu, Ziqi Liu, Li Wang, and Xiaolin Zheng. Vertically federated graph neural network for privacy-preserving node classification. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pp. 1959–1965, 2022a.

Jinyin Chen, Guohan Huang, Haibin Zheng, Shanqing Yu, Wenrong Jiang, and Chen Cui. Graph-fraudster: Adversarial attacks on graph neural network-based vertical federated learning. *IEEE Transactions on Computational Social Systems*, 10(2):492–506, 2022b.

Peng Chen, Jirui Yang, Junxiong Lin, Zhihui Lu, Qiang Duan, and Hongfeng Chai. A practical clean-label backdoor attack with limited information in vertical federated learning. In *2023 IEEE International Conference on Data Mining (ICDM)*, pp. 41–50. IEEE, 2023.

Peng Chen, Xin Du, Zhihui Lu, and Hongfeng Chai. Universal adversarial backdoor attacks to fool vertical federated learning. *Computers & Security*, 137:103601, 2024.

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 *31st USENIX Security Symposium (USENIX Security 22)*, pp. 1397–1414, 2022.

Yuhao Gu and Yuebin Bai. Lr-ba: Backdoor attack against vertical federated learning using local latent representations. *Computers & Security*, 129:103193, 2023.

Huilin He, Kun Zhu, Zewen Hu, Jie Wang, and Dawei Cheng. Deep reinforcement learning enhanced semi-supervised graph neural network for credit card fraud detection. In *Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence*, pp. 7201–7209, 2026.

Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. Unlearnable examples: Making personal data unexploitable. In *International Conference on Learning Representations*, 2021.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2):1–210, 2021. doi: 10.1561/2200000083.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Oscar Li, Jiankai Sun, Xin Yang, Weihao Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. Label leakage and protection in two-party split learning. In *International Conference on Learning Representations*, 2022.

Rui Liu, Pengwei Xing, Zichao Deng, Anran Li, Cuntai Guan, and Han Yu. Federated graph neural networks: Overview, techniques, and challenges. *IEEE transactions on neural networks and learning systems*, 36(3):4279–4295, 2024.

Yimin Liu, Peng Jiang, Qi Liu, and Liehuang Zhu. Generic adversarial attack framework against graph-based vertical federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 35662–35670, 2026.

Shiwei Lu, Ruihu Li, Wenbin Liu, and Xuan Chen. Defense against backdoor attack in federated learning. *Computers & Security*, 121:102819, 2022.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Peihua Mai and Yan Pang. Vertical federated graph neural network for recommender system. In *International conference on machine learning*, pp. 23516–23535. PMLR, 2023.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

Galileo Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. In *Proceedings of the Workshop on Mining and Learning with Graphs*, 2012.

Mohammad Naseri, Yufei Han, and Emiliano De Cristofaro. Badvfl: Backdoor attacks in vertical federated learning. In *2024 IEEE symposium on security and privacy (SP)*, pp. 2013–2028. IEEE, 2024.

Duy-Long Nguyen, Duc-Luong Ho-Viet, Anh-Thu Ngo-Tran, Quang H. Nguyen, and Binh P. Nguyen. MolHFCNet: Enhancing molecular graph representations with hierarchical feature combining and hybrid pretraining. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 9313–9321, 2025.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

Wei Shen, Wenke Huang, Guancheng Wan, and Mang Ye. Label-free backdoor attacks in vertical federated learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 20389–20397, 2025.

Yifan Sui, Yongqi Sun, Naiyue Chen, Yingying Zhao, Hongbo Cao, and Baomin Xu. Backdoor detection in federated learning with feature map: A multi-task learning perspective. *IEEE Transactions on Information Forensics and Security*, 2026.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Bo Wang, Yiming Yan, Maozhen Zhang, Wei Wang, and Hongwei Yao. Model backdoor attack on federated learning based on parameter analysis. *IEEE Transactions on Dependable and Secure Computing*, 2026.

Gang Wang, Qingming Li, Ximeng Liu, Xiaoran Yan, Qingkuan Dong, Huiwen Wu, Xiangjie Kong, and Li Zhou. Vfgcn: A vertical federated learning framework with privacy preserving for graph convolutional network. *IEEE Transactions on Dependable and Secure Computing*, 22(5):5617–5631, 2025.

Hanzhi Wang, Zhewei Wei, Junhao Gan, Sibo Wang, and Zengfeng Huang. Personalized pagerank to a target node, revisited. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 657–667, 2020.

Dongxian Wu and Yisen Wang. Adversarial neuron pruning purifies backdoored deep models. In *Advances in Neural Information Processing Systems*, volume 34, pp. 16913–16925, 2021.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 6861–6871, 2019.

Jirui Yang, Peng Chen, Zhihui Lu, Jianping Zeng, Qiang Duan, Xin Du, and Ruijun Deng. Backdoor attack on vertical federated graph neural network learning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 7885–7893, 2025.

Qinchen Yang, Zejun Xie, Hua Wei, Desheng Zhang, and Yu Yang. MalLight: Influence-aware coordinated traffic signal control for traffic signal malfunctions. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 2879–2889, 2024.

Razieh Nokhbeh Zaeem and K Suzanne Barber. The effect of the gdpr on privacy policies: Recent progress and future promise. *ACM Transactions on Management Information Systems (TMIS)*, 12 (1):1–20, 2020.

Oualid Zari, Chuan Xu, Javier Parra-Arnau, Ayşe Ünsal, and Melek Önen. Link inference attacks in vertical federated graph learning. In *2024 Annual Computer Security Applications Conference*, pp. 761–777. IEEE, 2024.

A ALGORITHM OF GRAPHCAMO

Algorithm 1 summarizes the training process of GraphCamo. The structure-guided trigger learning module uses local PPR to select support nodes at different hop distances and learns the corresponding shared triggers from gradients returned by the active party. In each training batch, the parties exchange embeddings and gradients through the standard VFGNN protocol, while training labels remain unchanged. The backdoor consolidation module uses a shared masked local graph and separate decoders to reconstruct clean and triggered features with the updated encoder. The trigger is added after masking and held fixed during reconstruction. Only the encoder's auxiliary gradient undergoes conflict projection and norm clipping.

Algorithm 1 GraphCamo

Require: Local graphs $\{G_i\}_{i=1}^{N}$; training labels; anchor node set $\mathcal{A}$.
Ensure: Learned triggers $\{\delta_k\}_{k=0}^{K}$ and model parameters Θ.

Initialize Θ, $\{\delta_k\}_{k=0}^{K}$, D_{rec}, and D_{trig}.
Adversary m**:**
    Use local PPR scores π_a to select and group support nodes into $\{\mathcal{S}_k\}_{k=0}^{K}$.
Perform clean warm-up with $\mathcal{L}_{\mathrm{cls}}$ and $\mathcal{L}_{\mathrm{rec}}$.
for each attack epoch **do**
    for each training batch $\mathcal{B}$ **do**
        Adversary m**:**
            Construct Δ from δ_k on $\mathcal{S}_k$ and inject it into X_m.
        Passive party:
            Compute and send $z_{i,v}$ to the active party; party m sends $\widetilde{z}_{m,v}$.
        Active party:
            Compute $\mathcal{L}_{\mathrm{cls}}$ and update θ_{top}.
            Return $q_{i,v}$ to each party i.
        Adversary m**:**
            Save g_{main} from local backpropagation.
        Passive party:
            Update θ_i using $q_{i,v}$.
        Adversary m**:**
            Update δ_k by PGD and rebuild Δ.
            Form X_m^{mask} using M and reuse the fixed E_m^{drop}.
            Compute $\mathcal{L}_{\mathrm{rec}}$ and $\mathcal{L}_{\mathrm{trig}}$.
            Obtain g_{aux} from the joint reconstruction objective $\mathcal{L}_{\mathrm{aux}}$.
            Correct g_{aux} using g_{main} when they conflict.
            Update θ_m, D_{rec}, and D_{trig} with their auxiliary gradients.
    end for
end for

B PROOFS OF THE THEORETICAL ANALYSIS

B.1 TRIGGER ATTENUATION UNDER GRAPH PROPAGATION

Proof of Proposition 1. Let A_m be the adjacency matrix of a finite, undirected, unweighted graph without self-loops, and define

$$\widetilde{A}_m = A_m + I, \qquad \widetilde{D}_m = \mathrm{diag}(\widetilde{A}_m \mathbf{1}), \qquad \widetilde{P}_m = \widetilde{D}_m^{-1} \widetilde{A}_m.$$

Every diagonal entry of $\widetilde{D}_m$ is positive, including at isolated nodes. The symmetric propagation operator satisfies $S_m = \widetilde{D}_m^{1/2} \widetilde{P}_m \widetilde{D}_m^{-1/2}$, and hence

$$H = S_m^L = \widetilde{D}_m^{1/2} \widetilde{P}_m^L \widetilde{D}_m^{-1/2}, \qquad H_{vv} = (\widetilde{P}_m^L)_{vv} \in [0, 1]. \tag{18}$$

The last assertion follows because $\widetilde{P}_m$ and $\widetilde{P}_m^L$ are row-stochastic. In particular, $(\widetilde{P}_m^L)_{vv}$ is the probability of returning to v after L random-walk steps. Unlike the induced-subgraph transition

matrix P_a used for PPR, $\widetilde{P}_m$ acts on the full local graph G_m with added self-loops. For $L = 1$, the return probability equals $1/(\deg_m(v) + 1)$, where $\deg_m(v)$ is the degree before adding self-loops.

Keep node features, representations, and perturbations as columns, with their transposes stored as matrix rows. A single-node perturbation is $\Delta = e_v \Delta_v^\mathsf{T}$, where e_v is the vth coordinate column vector. Linearity of the fixed bottom model f_m gives

$$\Delta Z_m = H\Delta W_m = H e_v \Delta_v^\mathsf{T} W_m.$$

Its row at node u is $H_{uv}\Delta_v^\mathsf{T} W_m$, so the column representation change is $\Delta z_{m,u} = H_{uv} W_m^\mathsf{T} \Delta_v$. The VFL counterpart $X_m W_m$ gives $W_m^\mathsf{T}\Delta_v$ at v and zero at other nodes. This proves Equation 14. For the active party's fixed linear top model f_g, write $F = Z_m B_m + F_{-m}$ and $\Gamma_{v,c} = F_{v,y_t} - F_{v,c}$ as in the proposition. The terms in F_{-m}, including all other parties' fixed contributions and the top model bias, do not change under this local perturbation. Thus

$$\Gamma^{\Delta}_{v,c} - \Gamma^{0}_{v,c} = H_{vv}\Delta_v^\mathsf{T} W_m \big(B_{m,:,y_t} - B_{m,:,c}\big),$$

which proves Equation 15. □

B.2 Reconstruction and Recoverable Feature Directions

Fix a training checkpoint of f_m, one of the existing branch decoders $D \in \{D_{\mathrm{rec}}, D_{\mathrm{trig}}\}$, the current perturbation Δ, and the dropped-edge graph with edge set E_m^{drop}. Sample a node uniformly from $\mathcal{B}$ for CR or $\mathcal{S}_{\mathrm{trig}}$ for TR, and independently sample a fresh feature mask according to Equation 8. The two branches may share each sampled mask, as in the algorithm. The sampled target x_{tar} is $x_{m,v}$ for CR and $x_{m,v} + \Delta_v$ for TR. Let $T = \mathrm{nrm}(x_{\mathrm{tar}})$, let Z be the corresponding random representation produced by f_m in Equation 9, and let $U = \mathrm{nrm}(D(Z))$. Here $\mathrm{nrm}(x) = x/\|x\|_2$ for $x \neq 0$ and $\mathrm{nrm}(0) = 0$. Conditioning on the fixed checkpoint avoids treating the decoder as a different estimator for each sampled example. The expectations below are over the sampled node and mask. Independence between graph nodes is not assumed. The results also apply to the empirical distribution of any fixed set of node–mask pairs, without implying generalization to other nodes or masks.

Proof of Proposition 2. When both vector norms are at least $\varepsilon_{\mathrm{num}}$, $\|T\|_2 = \|U\|_2 = 1$ and the numerical cosine equals $U^\mathsf{T}T$. Cauchy–Schwarz implies $U^\mathsf{T}T \leq 1$, so the outer nonnegative clamp does not change the loss. Therefore

$$\|T - U\|_2^2 = 2(1 - U^\mathsf{T}T), \qquad \mathbb{E}\|T - U\|_2^2 \leq 2\sqrt{\mathbb{E}(1 - U^\mathsf{T}T)^2} = 2\sqrt{R}.$$

Let $\mu(Z) = \mathbb{E}[T \mid Z]$. Since U is a measurable function of Z and T is bounded, conditional-expectation orthogonality gives

$$\mathbb{E}\|T - U\|_2^2 = \mathbb{E}\|T - \mu(Z)\|_2^2 + \mathbb{E}\|\mu(Z) - U\|_2^2,$$
$$\mathbb{E}\langle T - \mu(Z), \mu(Z) - U\rangle = 0.$$

Dropping the nonnegative second term proves Equation 16. The conditional mean is a mathematical Bayes estimator. Its appearance does not assume that the adversary can compute it or access any additional labels.

If T and Z are independent, T and U are also independent. With $\bar{t} = \mathbb{E}T$, Jensen's inequality and $\|\mathbb{E}U\|_2 \leq 1$ yield

$$R \geq (1 - \mathbb{E}[U^\mathsf{T}T])^2 = (1 - (\mathbb{E}U)^\mathsf{T}\bar{t})^2 \geq (1 - \|\bar{t}\|_2)^2.$$

The last step uses $(\mathbb{E}U)^\mathsf{T}\bar{t} \leq \|\bar{t}\|_2 \leq 1$. This proves Equation 17. □

The no-information threshold is a lower bound, not necessarily the optimal constant-decoder risk. Falling below it rules out independence for the evaluated distribution. Remaining above it does not imply independence. When target directions are nearly constant, the threshold approaches zero and is correspondingly uninformative. The recoverability result concerns directions, not feature magnitudes or Shannon mutual information.

Let $\mathcal{E}$ be the event that either the target or decoder-output norm is below $\varepsilon_{\mathrm{num}}$, let $p_{\mathrm{floor}} = \Pr(\mathcal{E})$, and let R now denote the expected loss actually used by the algorithm. Keep T and U defined by

nrm above. On $\mathcal{E}^c$, the preceding distance identity holds. On $\mathcal{E}$, $\|T - U\|_2^2 \le 4$, including when either vector is zero. Thus

$$\mathbb{E}\|T - \mathbb{E}[T \mid Z]\|_2^2 \le \mathbb{E}\|T - U\|_2^2 \le 2\sqrt{R} + 4p_{\mathrm{floor}}. \tag{19}$$

Indeed, apply Cauchy–Schwarz to the loss restricted to $\mathcal{E}^c$ and bound the remaining squared distance by $4p_{\mathrm{floor}}$. The ideal loss $(1 - U^{\mathsf{T}}T)^2$ and the implemented loss both lie in $[0, 4]$ and coincide outside $\mathcal{E}$. Under $T \perp Z$, the same Jensen argument remains valid for $\|T\|_2, \|U\|_2 \le 1$ and gives the conservative observed-risk bound

$$R \ge (1 - \|\mathbb{E}T\|_2)^2 - 4p_{\mathrm{floor}}. \tag{20}$$

If target norms are always above the floor and only decoder outputs are clipped, the sharper bound without the subtraction still holds: the implemented normalized output is a function of Z with norm at most one, so the same independence proof applies directly. These statements describe the analytical effect of norm floors. Ordinary floating-point roundoff should be handled by reporting numerical tolerances.

C Experimental Settings

C.1 Datasets

Cora and Cora_ml contain machine-learning papers represented by sparse word-based features (McCallum et al., 2000). Both have seven topic classes, but Cora_ml includes more citation links and a larger feature vocabulary than Cora. CiteSeer covers a broader range of computer-science publications. Its binary word features support classification into six research areas (Sen et al., 2008). PubMed instead focuses on diabetes-related medical literature. Its feature vectors contain TF–IDF word weights rather than word-presence indicators, and its three labels capture different diabetes-related topics (Namata et al., 2012). Cora, CiteSeer, and PubMed use 140, 120, and 60 training nodes, respectively. Each uses 500 validation nodes and $1{,}000$ test nodes. Cora_ml uses 280 training, 282 validation, and $2{,}248$ test nodes. Table 2 lists the size and feature dimension of each graph.

C.2 Local Models

We evaluate three local GNN architectures with different message-passing mechanisms. GCN aggregates neighboring features using normalized adjacency weights (Kipf & Welling, 2017). GAT assigns attention weights to neighbors so their contributions can differ (Veličković et al., 2018). SGC removes intermediate nonlinear transformations and uses simplified multi-hop propagation (Wu et al., 2019). Each party processes its local node features and edges with the selected bottom model. The active party concatenates the representations uploaded for the same node by all parties and applies its top model to predict the node class.

C.3 Attack Baselines and Graph Adaptation

Graph-Fraudster (GF) constructs an adversarial graph by querying intermediate representations and training a surrogate model (Chen et al., 2022b). TECB uses a few known target-class nodes and gradient alignment to strengthen the association between its trigger and the target class (Chen et al., 2023). VILLAIN combines label inference with a trigger injected into the malicious party's uploaded representations (Bai et al., 2023). Edge Triggering (ET) adds trigger edges between attack nodes and target-class nodes. Node Feature Triggering (NFT) inserts a fixed pattern into local node features, while Node Feature Replacement (NFR) substitutes attack-node features with the malicious party's features of target-class nodes. BVG injects triggers into the anchor nodes and their neighbors and uses representation similarity to retain the backdoor effect (Yang et al., 2025).

To apply VFL attacks to graph data, TECB and VILLAIN operate within the GNN models described in Appendix C.2. TECB inserts its feature pattern before graph propagation, whereas VILLAIN modifies the malicious party's embeddings after its bottom model. The graph trigger baselines operate on the malicious party's local nodes or edges.

GraphCamo and BVG use five clean warm-up epochs, a two-hop trigger range, and a trigger update step size of 0.2. NFT injects a repeating feature pattern with a trigger-strength factor of 4, while

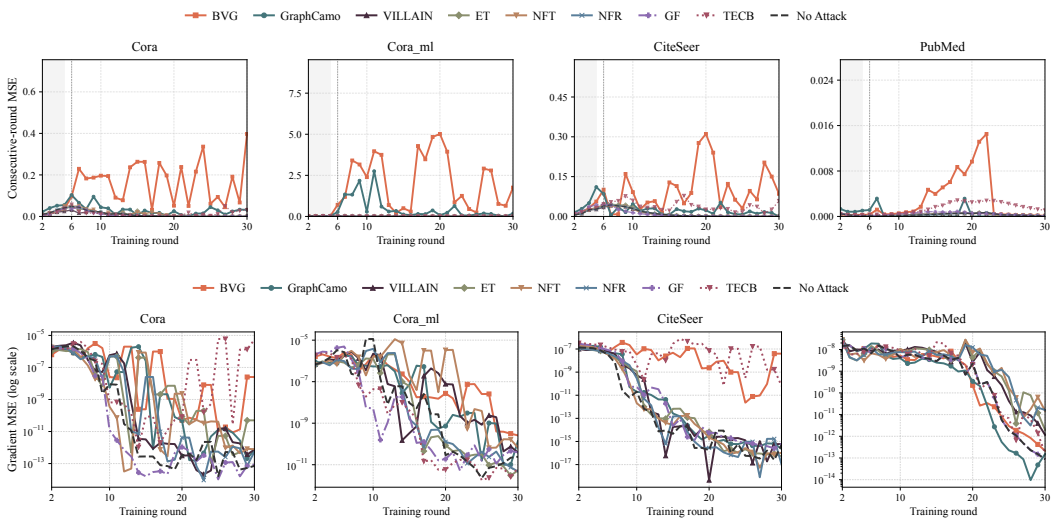


Figure 8: Embedding MSE (top) and gradient MSE (bottom) using GAT as the bottom model.

Table 2: Statistics of the experimental datasets.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Cora_ml	2,810	7,981	2,879	7
CiteSeer	3,327	4,732	3,703	6
PubMed	19,717	44,338	500	3

VILLAIN applies the corresponding pattern in the embedding space with the same strength factor. TECB uses the feature pattern with a trigger strength of 1 and replaces poisoned-node gradients with target-anchor gradients amplified by a factor of 5. At inference time, GraphCamo evaluates each attacked node v separately. For $k \in \{0, \ldots, K\}$, the learned δ_k is added to the malicious party's features at every node in $\mathcal{C}_{v,k} = \{u \in V : \mathrm{dist}_{G_m}(v, u) = k\}$. The original edges and the other parties' inputs remain unchanged.

C.4 Training Details and Hyperparameters

Training nodes are randomly shuffled in each epoch, and the model is updated with a batch size of 64. For each anchor node, PPR is computed on the malicious party's local graph with a damping factor of 0.70. In the masked reconstruction branch, the node-feature masking rate and local edge-drop rate are set to 0.15 and 0.10, respectively. The reconstruction-loss weight during clean warm-up is 0.10. During the attack stage, the weights of the clean feature reconstruction loss and trigger feature reconstruction loss are both 0.05. Both linear decoders use the Adam optimizer with a learning rate of 0.01. We apply one-sided gradient conflict projection only to the auxiliary gradient of the malicious bottom model and clip its global norm to 5.0. Each experiment is repeated five times.

C.5 Evaluation Metrics

We use attack success rate (ASR) and main-task accuracy (MTA) to evaluate attack effectiveness and clean-task performance, respectively (Bai et al., 2023; Yang et al., 2025; Gu & Bai, 2023). ASR is the fraction of non-target-class test nodes classified as the target class after trigger injection. MTA is the classification accuracy on test nodes without the trigger. Across the repeated runs, we report the mean and standard deviation, using ASR variation to assess attack reliability. To assess backdoor stealthiness, we compute the mean squared error (MSE) between consecutive epochs for uploaded embeddings and returned gradients separately. Smaller MSE values indicate less temporal change in these observable training signals. We also compare the curves with those from a no-attack setting.

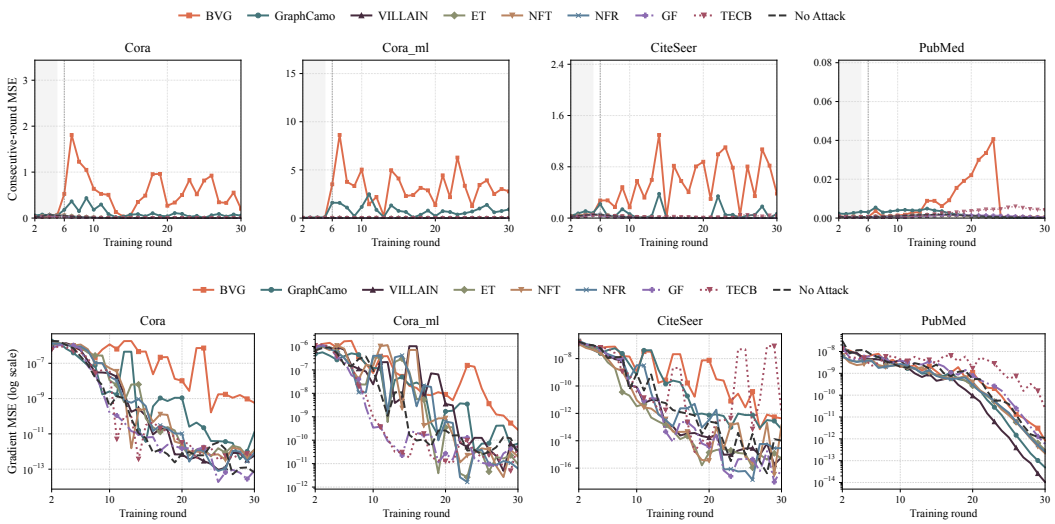


Figure 9: Embedding MSE (top) and gradient MSE (bottom) using SGC as the bottom model.

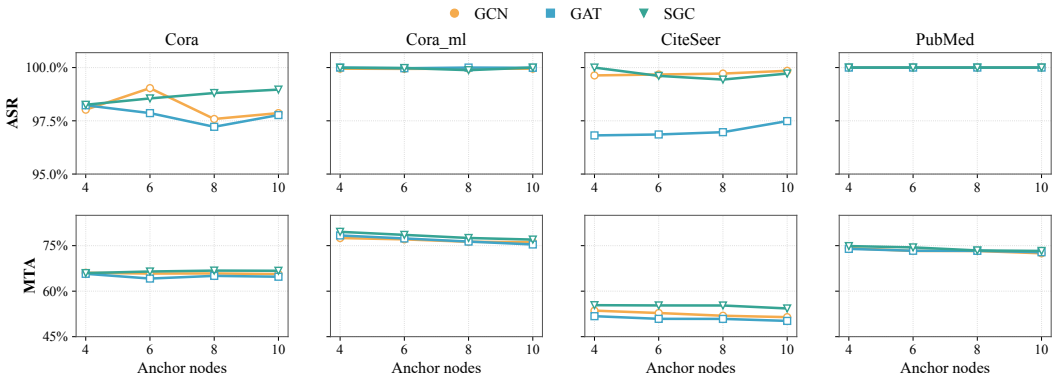


Figure 10: ASR and MTA of GraphCamo with 4, 6, 8, and 10 anchor nodes.

C.6 Defense Settings

ISO and DP-SGD inject Gaussian noise into uploaded embeddings and returned gradients, respectively, both with a norm-scaled noise ratio of 0.1. GC retains the largest-magnitude nonzero gradient components with a retention ratio of 0.1, whereas ANP prunes hidden neurons from the top model using a gradient-score threshold of 0.01.

D Further Experimental Analyses

D.1 Embedding and Gradient Stability

Figures 8 and 9 report the embedding and gradient stability results using GAT and SGC as the bottom models, respectively. For embedding stability, we record the embeddings last uploaded by the malicious party in each epoch. For gradient stability, we record the full-node gradients of the server's main-task loss with respect to those embeddings from the last minibatch of each epoch. After aligning nodes and embedding dimensions across epochs, we compute the MSE between consecutive epochs separately for each signal. The first epoch serves as the reference, so the curves begin at the second epoch. Gradient MSE is plotted on a logarithmic scale to show changes at different magnitudes.

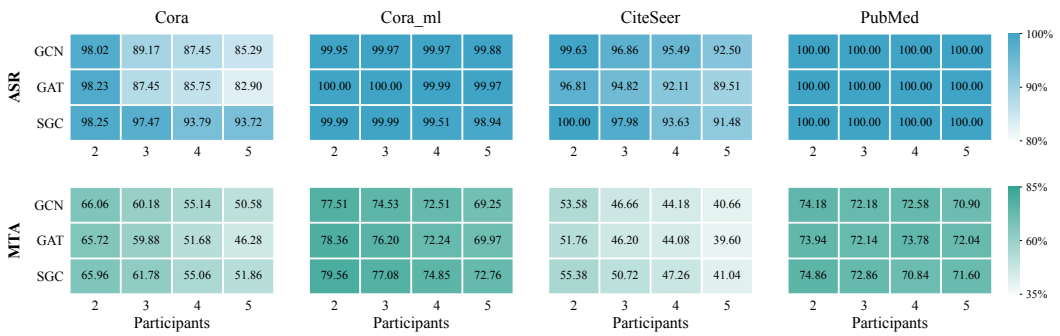


Figure 11: Results of GraphCamo with different numbers of participants, with one malicious participant in every setting. Heatmaps show mean ASR and MTA.

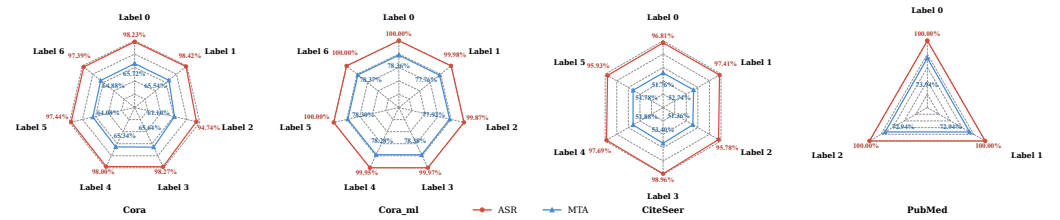


Figure 12: Results of GraphCamo with different target labels using GAT as the bottom model.

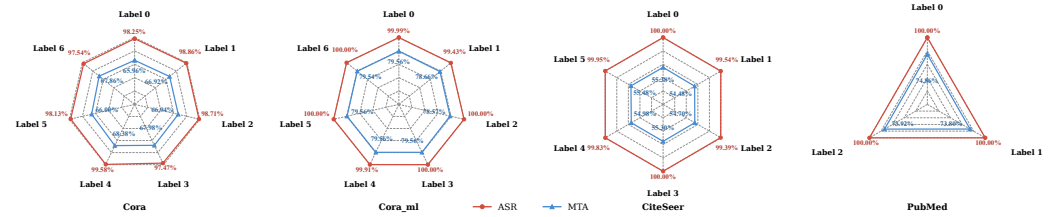


Figure 13: Results of GraphCamo with different target labels using SGC as the bottom model.

In the GCN results of Figures 4 and 5, GraphCamo's embedding MSE remains relatively low across the four datasets, whereas BVG exhibits pronounced attack-stage spikes. GraphCamo's gradient MSE generally declines and is below BVG's in many attack-stage rounds. Figures 8 and 9 show similar patterns for GAT and SGC. With GAT, GraphCamo has a few early embedding-MSE peaks, especially on Cora_ml, but BVG's peaks are larger and more persistent. With SGC, BVG again shows larger embedding-MSE spikes across the four datasets, while GraphCamo's peaks are smaller and shorter-lived. For both models, GraphCamo's gradient MSE generally decreases, whereas BVG exhibits more pronounced fluctuations on several datasets. Across GCN, GAT, and SGC, GraphCamo's gradient trajectory on PubMed broadly follows the no-attack baseline. These results show that GraphCamo is less detectable.

D.2 Effect of Number of Anchor Nodes

Figure 10 shows that increasing the number of anchor nodes does not yield a consistent ASR gain. On Cora_ml and PubMed, ASR stays near 100% across all tested counts and bottom models. On Cora, the GCN and GAT curves fluctuate slightly, while on CiteSeer the GAT results remain below those of GCN and SGC. MTA is broadly stable on Cora but tends to decrease modestly as more anchor nodes are used on Cora_ml, CiteSeer, and PubMed. Thus, 4 anchor nodes already provide strong attack effectiveness in these settings, whereas adding more anchors offers no systematic benefit and can slightly reduce clean-task performance. A larger anchor set may also expand the neighborhood affected during trigger injection.

D.3 Effect of Number of Participants

Figure 11 compares settings with two to five participants and one malicious participant. On Cora_ml, ASR remains at least 98.94% across GCN, GAT, and SGC, while PubMed maintains 100% ASR in every setting. In contrast, Cora's ASR with GAT falls from 98.23% to 82.90%, and CiteSeer's ASR with GAT falls from 96.81% to 89.51% as the participant count rises from two to five. This is possibly because representations from additional parties dilute the malicious party's influence.

D.4 Performance Across Target Labels

Figures 12 and 13 extend the target-label analysis to GAT and SGC. With GAT, ASR exceeds 95% for all but one setting: target label 2 on Cora yields 94.74%. With SGC, ASR is at least 97.47% in every evaluated setting. Both models reach 100% ASR for every target label on PubMed. MTA varies by only a few percentage points across labels within each dataset. The largest spread occurs for GAT on Cora, from 61.10% to 65.72%. These results support the conclusion that GraphCamo's attack effectiveness is not restricted to a particular target class.