A Robust Detection and Correction Framework for GNN-Based Vertical Federated Learning

Zhicheng Yang[1,2], Xiaoliang Fan[1,2(✉)], Zheng Wang[1,2], Zihui Wang[1,2], and Cheng Wang[1,2]

[1] Fujian Key Laboratory of Sensing and Computing for Smart Cities, School of Informatics, Xiamen University, Xiamen 361005, People's Republic of China
{zcyang,zwang,wangziwei}@stu.xmu.edu.cn, {fanxiaoliang,cwang}@xmu.edu.cn
[2] Key Laboratory of Multimedia Trusted Perception and Efficient Computing, Ministry of Education of China, Xiamen University, Xiamen 361005, People's Republic of China

Abstract. Graph Neural Network based Vertical Federated Learning (GVFL) facilitates data collaboration while preserving data privacy by learning GNN-based node representations from participants holding different dimensions of node features. Existing works have shown that GVFL is vulnerable to adversarial attacks from malicious participants. However, how to defend against various adversarial attacks has not been investigated under the non-i.i.d. nature of graph data and privacy constraints. In this paper, we propose RDC-GVFL, a novel two-phase robust GVFL framework. In the detection phase, we adapt a Shapley-based method to evaluate the contribution of all participants to identify malicious ones. In the correction phase, we leverage historical embeddings to rectify malicious embeddings, thereby obtaining accurate predictions. We conducted extensive experiments on three well-known graph datasets under four adversarial attack settings. Our experimental results demonstrate that RDC-GVFL can effectively detect malicious participants and ensure a robust GVFL model against diverse attacks. Our code and supplemental material is available at https://github.com/zcyang-cs/RDC-GVFL.

Keywords: GNN-based Vertical Federated Learning · Adversarial attack · Robustness

1 Introduction

Graph Neural Network (GNN) has gained increasing popularity for its ability to model high-dimensional feature information and high-order adjacent information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-981-99-8435-0_8.

Q. Liu et al. (Eds.): PRCV 2023, LNCS 14427, pp. 97–108, 2024.
https://doi.org/10.1007/978-981-99-8435-0_8

on graphs [17]. In practice, privacy constraints prevent GNNs from learning better node representations when node features are held by different participants [6]. To enable collaborative improvement of node representation while protecting data privacy, GNN-based Vertical Federated Learning (GVFL) [2,3,12] employs GNNs locally for representation learning and Vertical Federated Learning (VFL) globally for aggregated representation.

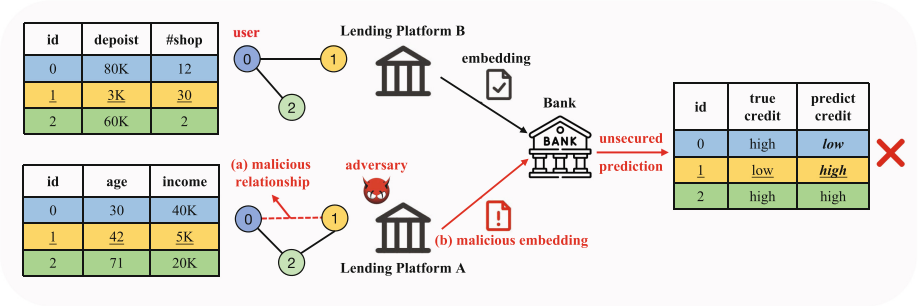

Fig. 1. An motivating example: an adversary can conduct various adversarial attacks: (a) manipulate the local connection relationship for fraudulent loan purposes by adding a connection from a low-credit user (user 1) to a high-credit user (user 0); and (b) upload the malicious embedding to deliberately misclassify the bank. To mitigate lending risks, the bank must implement effective defense measures.

In real-world applications, GVFL model is vulnerable to adversarial attacks [3]. For example, in Fig. 1, The adversary could compromise lending platform A and modify the uploaded embedding information to manipulate the bank's eventual prediction of the user's credit. Therefore, it is crucial for the bank to incorporate robust defensive measures, as failure to do so could result in loans being granted to users with lower credit.

However, neither graph defenses nor federated learning defenses against adversarial attacks can be directly applied in GVFL. On one hand, privacy restrictions prevent graph defenders from accessing training data and models in GVFL, which are essential for effectively implementing defense mechanisms [7,8]. On the other hand, federated learning defenses encounter challenges when dealing with inter-dependencies among node embeddings within a graph structure. Existing approaches, such as Copur's robust autoencoder [10,11], assume no dependency between embeddings of each sample, but the presence of inter-dependencies among node embeddings poses a hurdle for defenders to accurately learn the appropriate feature subspace.

To address the issues above, we propose a novel *Robust Detection and Correction Framework for GNN-based Vertical Federated Learning* (RDC-GVFL) to enhance the robustness of GVFL. RDC-GVFL has a detection phase and prediction phase. **In the detection phase**, the server first requires each participant to generate local embeddings to upload. Then, the server computes

each participant's contribution to the validation set based on the Shapley Value, and identifies the participant with the lowest contribution as the malicious participant. **In the correction phase**, the server retrieves the relevant embeddings from historical embedding memory to correct malicious embeddings. Extensive experiments on three datasets against four adversarial attack settings confirm the effectiveness of RDC-GVFL.

The main contributions of our work can be summarized as follows:

- We propose RDC-GVFL, a novel robust framework for GVFL, which can enhance the robustness of GVFL under various attack scenarios. To the best of our knowledge, this is the first work dedicated to defending against adversarial attacks in GVFL.
- We present a Shapley-based detection method, enabling the effective detection of malicious participants in GVFL. Additionally, we propose a correction mechanism that utilizes historical embeddings to generate harmless embeddings, thereby obtaining accurate predictions.
- We conduct extensive experiments on three well-known graph datasets under four types of adversarial attacks. Experimental results demonstrate the effectiveness of the RDC-GVFL for a robust GVFL system.

2 Related Works

2.1 Attack and Defense in Graph Neural Networks

Extensive studies have demonstrated that GCNs are vulnerable to adversarial attacks [17]. These attacks can be classified into two categories based on the attack stage and goal. Evasion attacks (test-time) aim to deceive the GCN during inference, and they have been studied in works such as [20]. On the other hand, poisoning attacks (training-time) occur during the training process and aim to manipulate the GCN's learned representations, as explored in works like [19]. To enhance the robustness of GCNs, many defenses are proposed and they can be classified into three categories including improving the graph [8], improving the training [5], and improving the architecture [4] based on their strategy. However, all of these existing defenses require the defender to inspect either the training data or the resulting model, which is impractical in GVFL.

2.2 Attack and Defense in Vertical Federated Learning

Existing attacks on VFL have been shown to undermine model robustness [13]. For instance, the passive participant can launch adversarial attacks with different objectives. In targeted attacks, specific labels are assigned to triggered samples [14]. On the other hand, in non-targeted attacks, noise is added to randomly selected samples, or missing features are introduced to impair the model's utility [11]. For defense, RVFR [10] and Copur [11] is designed to defend against adversarial attacks. These defense approaches utilize feature purification techniques. However, they can not be directly applied to GVFL since the inter-dependencies among node embeddings in a graph structure makes it challenging to learn an appropriate feature subspace for defense purpose.

2.3 Attack and Defense in GNN-Based Vertical Federated Learning

GVFL has been found to be vulnerable to adversarial attacks, as highlighted in recent studies. One such attack method, called Graph-Fraudster, was proposed to perform evasion attacks in GVFL [3]. This attack assumes that a malicious participant can infer additional embedding information of a normal participant from the server. By inferring a perturbed adjacency matrix from normal embedding, the adversary can generate malicious embeddings. In practice, the adversary can employ various attack vectors to threaten GVFL and the defense against adversarial attacks in GVFL remains an open research issue.

3 Methodology

In this section, We first describe the GVFL system and threat models of GVFL. Next, we provide an overview of the proposed RDC-GVFL framework. Finally, we present the detection and correction methods of our framework. For convenience, the definitions of symbols used in this paper are listed in the Section A of Supplemental Material.

3.1 GNN-Based Vertical Federated Learning

As described in [2], GVFL involves M participants and a server that collaboratively train a model based on a graph dataset $G = \{G_1, ..., G_M, Y\}$. Here, $Y = \{y_j\}_{j=1}^{N}$ denotes labels of $|N|$ nodes held by the server, and $G_i = (V, E_i, \mathbf{X}_i)$ represents the subgraph held by participant i, where $\mathbf{X}_i = \{\mathbf{x}_j^i\}_{j=1}^{N}$ denotes the features associated with nodes V and edges E_i. The training process of GVFL is similar to VFL. Firstly, each participant i extract features and obtains the local embedding $\mathbf{h}_j^i = f_i(A_i, \mathbf{X}_i; \theta_i)$ of node j using its local Graph Neural Network f_i. Then, the server aggregates the node embeddings from all participants and generates the label prediction $\hat{y}_j$ for node j using its global classifier f_0.

3.2 Threat Model

Adversary. There are M participants which possess distinct subsets of features from the same set of N training instances, along with one adversary attempting to compromise the GVFL system. For simplicity, we assume the adversary is one of the participants in GVFL, referred to as the malicious participant.

Adversary's Goal. The malicious participant aims to induce the GVFL model to predict the wrong class. This may serve the malicious participant's interests, such as earning profits by offering a service that modifies loan application results at a bank or disrupting the normal trading system.

Adversary's Capability. We assume the training process is well-protected (attack-free). However, during the inference phase, the malicious participant can manipulate local data to send specific poisoning-embedded features to the server or send arbitrary embedded features to the server. As described in [3,11], we characterize the malicious participant's attack vector into the following four types and assume each malicious participant performs the same attack per round:

(a) **Graph-Fraudster attack** [3]: The malicious participant first steals normal embeddings from other participants. Then, the malicious participant adds noise to the stolen embeddings and computes an adversarial adjacency matrix $\hat{A}$ by minimizing the Mean Squared Error loss between the embedded features $f(\hat{A}, \mathbf{x})$ and the noise-added embedding $\mathbf{h}^m$. Finally, the malicious participant sends the poisoning embedded features based on the adversarial adjacency matrix.

$$\hat{A} = \arg\min_{\hat{A}} MSE(f(\hat{A}, \mathbf{x}), \mathbf{h}^m) \tag{1}$$

$$h_{gf} = f(\hat{A}, \mathbf{x}) \tag{2}$$

(b) **Gaussian feature attack**: The malicious participant sends malicious feature embedding which is added by Gaussian noise.

$$\mathbf{h}_{gauss} = \mathbf{h} + \mathcal{N}(\mu, \mathbf{\Sigma}) \tag{3}$$

(c) **Missing feature attack** [11]: The malicious participant don't send any embedded feature to the server. This can occur when the participant's device is damaged or disconnected.

$$\mathbf{h}_{miss} = \mathbf{0} \tag{4}$$

(d) **Flipping feature attack** [11]: The malicious participant sends adversarial feature embedding $\mathbf{h}_{flip}$ to mislead the server, whose magnitude λ can be arbitrarily large.

$$\mathbf{h}_{flip} = -\mathbf{h} * \lambda \tag{5}$$

3.3 Framework Overview

As shown in Fig. 2, we propose a two-phase framework called RDC-GVFL to enhance the robustness of GVFL. The Detection phase takes place between the model training and inference phases. During the Detection phase, the server collects the embeddings of participants from the validation dataset. Subsequently, the server calculates the contribution of each participant and identifies the one with the lowest contribution as the malicious participant. The Correction phase replaces the original model inference phase, and the server maintains an embedding memory that stores the embeddings of all training nodes. In this phase, the server utilizes normal embedding set as a query to retrieve the most similar embedding from historical embedding memory to correct the malicious embedding, thereby obtaining accurate predictions and sending them to all participants. The pseudo-code for RDC-GVFL is given in Algorithm 1.

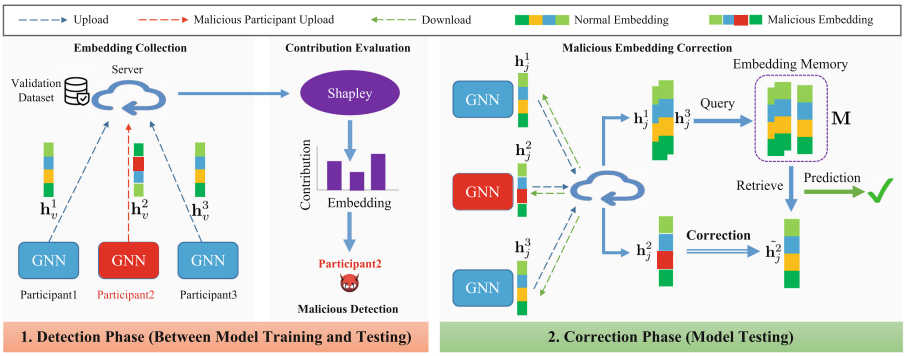

Fig. 2. Overview of the RDC-GVFL framework including two phases: (a) detection phase: the server using a validation dataset to evaluate contribution and identify the malicious participant; and (b) correction phase: the server leveraging the historical embeddings to retrieve the relevant embedding to correct the malicious embedding for accurate predictions sent to all participants.

3.4 Malicious Participant Detection

In addition to the training and testing processes of GVFL, we introduce a Detection phase to detect the malicious participant. In this phase, we propose a Shapley-based method to detect the malicious participant and it consists of three stages: embedding collection, contribution evaluation and malicious participant detection.

(a) **Embedding collection.** The server maintains a validation dataset $\mathcal{D}_{val} = \{V_{val}, Y_{val}\}$, which is used to mandate each participant generates a local embedding and sends it to the server. A normal participant i will send the embedding $\mathbf{h}_v^i$ of node $v \in V_{val}$ to the server while the malicious participant m, unaware of the server's detection process, will upload a poisoning embedding $\mathbf{h}_v^m$. As a result, the server collects a set of local embeddings aligned with its validation set.

(b) **Contribution evaluation.** We leverage Shapley Value, a fair contribution valuation metric derived from cooperative game theory, to evaluate contribution in GVFL. Specifically, we define the value function of a set of embeddings $F_v(S)$, which indicates whether the correct classification of node v can be achieved by making prediction using only the set S. The server then considers the total marginal value of each embedding $\mathbf{h}_v^i$ across all possible sets S as the contribution of participant i. This can be formalized as follows:

$$F_v(S) = \mathbb{I}(f_0(Agg(S), \theta_0) = y_v) \tag{6}$$

$$\phi_v^i(\{\mathbf{h}_v^i\}_{i=1}^M) = \sum_{S \subseteq (\mathbf{h}_v^1, \ldots \mathbf{h}_v^M) \setminus \mathbf{h}_v^i} \frac{|S|!(M - |S| - 1)!}{M!} (F_v(S \cup \mathbf{h}_v^i) - F_v(S)) \tag{7}$$

(c) **Malicious participant detection.** After the embedding collecting stage and contribution computation stage, the server accumulates the contributions of each participant on the validation set and identifies the participant with the lowest total contribution as a potential malicious participant $\hat{m}$. Based on this identification, the server may impose penalties on the malicious participant. This can be formalized as follows:

$$\hat{m} = \arg\min_{i} \sum_{v=1}^{|\mathcal{D}_{val}|} \phi_v^i(\{\mathbf{h}_v^i\}_{i=1}^M) \tag{8}$$

3.5 Malicious Embedding Correction

In the Correction phase, we leverage historical node embedding for correcting malicious embedding, which relies on the inter-dependency between nodes in the inference phase and nodes in the training phase. Specifically, during this phase, the server is already aware of the identity of the malicious participant, and it maintains a node embedding memory $\mathbf{M} \in \mathbb{R}^{N \times M}$ that stores the embeddings of all training nodes from all participants. When the GVFL model performs inference, i.e. each participant i sends its embedding $\mathbf{h}_j^i$ of node j to the server requesting prediction, the server will utilizes normal embedding set $\{\mathbf{h}_j^k\}$ where $k \in [M] \setminus m$ as a query to retrieve the most similar embedding $\mathbf{h}_\ell^m$ from the node embedding memory $\mathbf{M}$. Subsequently, the malicious embedding $\mathbf{h}_j^m$ is corrected or adjusted according to the $\mathbf{h}_\ell^m$. This process can be formulated as follows:

$$\ell = \arg\max_{\ell \in [N]} \{ \sum_{k \in [M] \setminus m} \frac{\mathbf{h}_j^k \cdot \mathbf{h}_\ell^k}{||\mathbf{h}_j^k|| \cdot ||\mathbf{h}_\ell^k||} \} \tag{9}$$

$$\mathbf{h}_j^m \leftarrow \mathbf{h}_\ell^m \tag{10}$$

4 Experiment

In this section, we carefully conduct comprehensive experiments to answer the following three research questions.

RQ1 Does the Shapley-based detection method within the RDC-GVFL framework have the capability to detect malicious participants?

RQ2 How does the performance of the RDC-GVFL framework compare to that of other defense methods?

RQ3 How does discarding the malicious embedding directly without correcting it affect the robustness?

4.1 Experiment Settings

Datasets. We use three benchmark datasets, i.e., Cora, Cora_ML [15] and Pubmed [16]. Dataset statistics are summarized in Table 1. The details of partition and evaluation strategies are described in the Section B of Supplemental Material.

Algorithm 1: RDC-GVFL

Input : Number of participants M, Validation Dataset $\mathcal{D}_{val}$, input embedding matrix $\mathbf{H}_j$ of node j
Output: Prediction label $\hat{y}_j$ of node j
`// Model Training`
Train the GVFL model to obtain the server model f_0 and local GNN models f_i;
Retrain the GVFL model to maintain the embedding memory $\mathbf{M}$;
$m = \textbf{ServerDetection}(\mathcal{D}_{val})$;
`// Model Inference`
$\hat{y}_j = \textbf{ServerCorrection}(m, \mathbf{M}, \mathbf{H}_j)$;

ServerDetection($\mathcal{D}_{val}$):
Initialize contribution list $C = [0] * M$;
for *v = 1, ..., $|\mathcal{D}_{val}|$* **do**
  Randomly Sample $S \subset [|\mathcal{D}_{val}|]$;
  for *each participant i in parallel* **do**
    if *i belongs to malicious participant* **then**
      Malicious i computes and sends $\{\overline{\mathbf{h}_k^i}\}_{k \in S}$ to server accroding to (1)-(5);
    else
      participant i computes and sends $\{\mathbf{h}_k^i\}_{k \in S}$ to server;
  Server computes contribution c_i of each participant i according to (6)-(7);
  for $i = 1, ..., M$ **do**
    $C[i] \leftarrow C[i] + c_i$;
$m \leftarrow \arg\min_i C$;
return m;

ServerCorrection(m, $\mathbf{M}$, $\mathbf{H}_j$):
$\mathbf{H}_j = [\mathbf{h}_j^1 ... \mathbf{h}_j^m ... \mathbf{h}_j^M]$;
Server using normal embedding looks up in the node embedding memory $\mathbf{M}$ to retrieve the most relevant embedding $\mathbf{h}_\ell^m$ according to (9)-(10);
$\mathbf{h}_j^m \leftarrow \mathbf{h}_\ell^m$;
Server gets the prediction $\hat{y}_j = f_0(Agg(\mathbf{h}_j^1, ..., \mathbf{h}_j^m, ..., \mathbf{h}_j^M))$;
return $\hat{y}_j$;

Baseline Methods. We compare with 1) **Unsecured**: no defense. 2) **Krum** [1]: As no prior research has focused on robust GVFL, We borrow the idea of Krum which is the most well-known robust algorithm in FL to identify the participant with the highest pair-wise distance as the malicious participant, and discard its embedding for defense. 3) **RDC w/o Cor.**: We employ the RDC-GVFL framework to detect malicious participants, and subsequently, we directly discard the embedding associated with the identified malicious participant as a defense measure. 4) **Oracle**: where there is no attack. We use GCN [9] and SGC [18] as the local GNN model, the details of these models and parameter settings are shown in the Section B of Supplemental Material.

Table 1. The statistics of datasets.

Datasets	#Nodes	#Edges	#Features	#Classes	#Average Degree
Cora	2708	5429	1433	7	2.00
Cora_ML	2810	7981	2879	7	2.84
Pubmed	19717	44325	500	3	2.25

Table 2. Detection Rate(%) / Test Accuracy(%) against different attacks. 'V1/V2' represents scenarios where detection rate achieved 100% while 'V1/V2' represents scenarios where detection rate is below 100%.

Dataset	Attack	Local Model					
		GCN			SGC		
		M=2	M=3	M=4	M=2	M=3	M=4
Cora	Oracle	−−/80.13	−−/79.97	−−/78.97	−−/77.90	−−/77.73	−−/77.53
	GF	100/67.80	100/70.00	91.7/72.93	100/61.37	100/65.17	100/68.57
	Gaussian	100/69.93	89.0/68.97	83.3/69.00	100/37.20	100/52.97	100/54.97
	Missing	100/52.97	100/65.53	100/67.30	100/63.10	100/70.33	100/72.03
	Flipping	100/20.50	100/33.50	100/42.80	100/21.93	100/41.17	100/50.70
Cora_ML	Oracle	−−/84.90	−−/84.93	−−/84.87	−−/83.30	−−/83.30	−−/83.27
	GF	100/72.30	78.0/74.77	100/76.23	100/65.33	100/71.57	100/74.37
	Gaussian	100/64.47	78.0/63.50	100/62.93	100/35.73	100/35.57	100/35.87
	Missing	100/53.50	100/63.80	100/73.97	100/72.20	100/76.00	100/77.97
	Flipping	100/24.93	100/39.33	100/46.30	100/24.03	100/42.1	100/53.53
Pubmed	Oracle	−−/77.83	−−/78.33	−−/77.63	−−/75.80	−−/76.17	−−/75.63
	GF	100/44.13	100/53.17	100/51.80	100/35.30	100/36.97	100/42.47
	Gaussian	100/59.10	100/57.60	91.7/56.77	100/43.13	100/43.10	100/43.30
	Missing	100/60.57	100/62.47	100/67.13	100/64.37	100/68.33	100/67.40
	Flipping	100/35.03	100/42.63	100/51.97	100/34.77	100/51.80	100/55.67

4.2 Detection Performance(RQ1)

To answer **Q1**, we measured the detection rate of our detection method under the various attack settings, where the detection rate is calculated as the ratio of detected malicious participants to the total number of potential cases of malicious participants. We summarized the results in Table 2. Based on this results, we can draw several conclusions.

- **RDC-GVFL can effectively detects malicious participants across different attack scenarios**: Our detection method exhibits a high detection rate for identifying malicious participants in diverse scenarios. Specifically, our method achieves a 100% recognition rate against Missing attack and Flipping attack, while in other attack scenarios, the recognition rate remains above 78%.
- **RDC-GVFL is more effective against more significant attacks**: The effectiveness of identification increases as the attacks become more significant. For example, GF and Gaussian attacks exhibit higher effectiveness when the local model is SGC, and RDC-GVFL is capable of identifying the malicious participant in all these scenarios.

To better illustrate the effectiveness of the detection method, we visualize the contribution of each participant before and after the attack. These results can be found in Section C of Supplemental Material.

Table 3. The accuracy (%) of each defense under five adversarial attack scenarios on multiple datasets with a varying number of participants.

Local Model	Dataset	M	Attack	Defense			
				Unsecured	Krum	Ours w/o Cor.	Ours
GCN	Cora	2	Oracle	**80.10**	53.79	54.10	70.93
			GF	67.80	39.70	52.99	**70.76**
			Gaussian	69.93	47.60	52.99	**70.76**
			Missing	52.97	32.50	52.99	**70.76**
			Flipping	20.50	28.01	52.99	**70.76**
		3	Oracle	**80.00**	64.80	64.80	74.26
			GF	70.00	52.97	65.54	**74.23**
			Gaussian	68.97	65.54	64.20	**72.54**
			Missing	65.53	49.16	65.54	**74.23**
			Flipping	33.50	32.47	65.54	**74.23**
		4	Oracle	**79.00**	62.36	68.53	73.33
			GF	72.93	56.68	67.31	**73.86**
			Gaussian	69.00	67.32	66.29	**72.61**
			Missing	67.30	53.18	67.32	**74.35**
			Flipping	42.80	40.63	67.32	**74.35**
	Cora_ML	2	Oracle	**84.90**	51.55	56.59	78.09
			GF	72.30	42.75	53.46	**77.63**
			Gaussian	64.47	44.95	53.76	**71.10**
			Missing	53.50	32.17	53.46	**77.63**
			Flipping	24.93	28.67	53.46	**77.63**
		3	Oracle	**84.93**	54.41	71.60	82.29
			GF	74.77	47.27	64.45	**75.19**
			Gaussian	63.50	63.76	65.46	**75.00**
			Missing	63.80	32.15	63.76	**79.89**
			Flipping	39.33	23.64	63.76	**79.89**
		4	Oracle	**84.87**	69.64	76.89	82.69
			GF	76.23	60.43	74.07	**82.06**
			Gaussian	62.93	74.07	74.07	**82.06**
			Missing	73.97	53.56	74.07	**82.06**
			Flipping	46.30	28.64	74.07	**82.06**
	Pubmed	2	Oracle	**77.83**	61.00	64.20	71.56
			GF	44.13	46.68	60.35	**71.85**
			Gaussian	59.10	53.35	60.35	**71.85**
			Missing	60.57	47.26	60.35	**71.85**
			Flipping	35.03	37.18	60.35	**71.85**
		3	Oracle	**78.33**	56.43	70.00	74.76
			GF	53.17	54.30	62.29	**72.78**
			Gaussian	57.60	62.29	62.29	**72.78**
			Missing	62.47	54.70	62.29	**72.78**
			Flipping	42.63	34.02	62.29	**72.78**
		4	Oracle	**77.63**	63.76	70.70	76.16
			GF	51.80	56.50	67.66	**73.17**
			Gaussian	56.77	67.66	67.47	**72.15**
			Missing	67.13	57.27	67.66	**73.17**
			Flipping	51.97	41.74	67.66	**73.17**

4.3 Defense Performance(RQ2-RQ3)

To answer **Q2-Q3**, we conducted a series of experiments with different defense methods against various attack scenarios. The results based on GCN are summarized in Tabel 3 and the other results based on SGC can be found in Section D of Supplemental Material. Observations are concluded in this experiments.

- **RDC-GVFL exhibits a higher level of robustness compared to other defense methods**: RDC-GVFL consistently outperforms unsecured and Krum under various attack scenarios. For instance, on the GVFL based on two participants with Cora dataset, when a malicious participant performs Flipping attack, our RDC-GVFL framework demonstrates a remarkable accuracy improvement from 20.50% to 70.76%, while Krum only manages to enhance the accuracy to 28.01% from the same starting point of 20.50%.
- **Discarding malicious embeddings harms GVFL**: RDC-GVFL without correction does not generally improve accuracy compared to unsecured. This observation suggests that malicious embeddings contain information useful for model prediction, emphasizing the importance of their correction.
- **Negligible loss of accuracy with RDC-GVFL**: In the absence of an adversary, the use of the RDC-GVFL framework results in a maximum accuracy degradation of less than 10%. However, in the presence of an adversary, the RDC-GVFL framework can significantly enhance model accuracy and ensure the robustness of the GVFL system.

5 Conclusion

In this work, we introduced a novel robust framework of GVFL integrated with the Shapley-based detection and the correction mechanism to effectively defend against various attacks during inference in GVFL. In addition, the proposed detection method may have much wider applications due to the post-processing property. Through extensive experiments, we have demonstrated that our framework can achieve high detection rates for malicious participants and enhance the robustness of GVFL. For future work, we plan to extend our method to more scenarios and explore more effective defense methods for robust GVFL.

Acknowledgement. This work was supported by Natural Science Foundation of China (62272403, 61872306), and FuXiaQuan National Independent Innovation Demonstration Zone Collaborative Innovation Platform (No.3502ZCQXT2021003).

References

1. Blanchard, P., El Mhamdi, E.M., Guerraoui, R., Stainer, J.: Machine learning with adversaries: byzantine tolerant gradient descent. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
2. Chen, C., et al.: Vertically federated graph neural network for privacy-preserving node classification. arXiv preprint arXiv:2005.11903 (2020)

3. Chen, J., Huang, G., Zheng, H., Yu, S., Jiang, W., Cui, C.: Graph-Fraudster: adversarial attacks on graph neural network-based vertical federated learning. IEEE Trans. Comput. Soc. Syst. **10**, 492–506 (2022)
4. Chen, L., Li, J., Peng, Q., Liu, Y., Zheng, Z., Yang, C.: Understanding structural vulnerability in graph convolutional networks. arXiv preprint arXiv:2108.06280 (2021)
5. Feng, W., et al.: Graph random neural networks for semi-supervised learning on graphs. Adv. Neural. Inf. Process. Syst. **33**, 22092–22103 (2020)
6. Fu, X., Zhang, B., Dong, Y., Chen, C., Li, J.: Federated graph machine learning: a survey of concepts, techniques, and applications. ACM SIGKDD Explor. Newsl. **24**(2), 32–47 (2022)
7. Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., Tang, J.: Graph structure learning for robust graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 66–74 (2020)
8. Jin, W., Zhao, T., Ding, J., Liu, Y., Tang, J., Shah, N.: Empowering graph representation learning with test-time graph transformation. arXiv preprint arXiv:2210.03561 (2022)
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
10. Liu, J., Xie, C., Kenthapadi, K., Koyejo, S., Li, B.: RVFR: robust vertical federated learning via feature subspace recovery. In: NeurIPS Workshop New Frontiers in Federated Learning: Privacy, Fairness, Robustness, Personalization and Data Ownership (2021)
11. Liu, J., Xie, C., Koyejo, S., Li, B.: CoPur: certifiably robust collaborative inference via feature purification. Adv. Neural. Inf. Process. Syst. **35**, 26645–26657 (2022)
12. Liu, R., Xing, P., Deng, Z., Li, A., Guan, C., Yu, H.: Federated graph neural networks: Overview, techniques and challenges. arXiv preprint arXiv:2202.07256 (2022)
13. Liu, Y., et al.: Vertical federated learning. arXiv:2211.12814 (2022)
14. Liu, Y., Yi, Z., Chen, T.: Backdoor attacks and defenses in feature-partitioned collaborative learning. arXiv preprint arXiv:2007.03608 (2020)
15. McCallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Inf. Retrieval **3**, 127–163 (2000)
16. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93–93 (2008)
17. Wu, B., et al.: A survey of trustworthy graph learning: reliability, explainability, and privacy protection. arXiv preprint arXiv:2205.10014 (2022)
18. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: International conference on machine learning, pp. 6861–6871. PMLR (2019)
19. Zhang, S., Chen, H., Sun, X., Li, Y., Xu, G.: Unsupervised graph poisoning attack via contrastive loss back-propagation. In: Proceedings of the ACM Web Conference 2022, pp. 1322–1330 (2022)
20. Zügner, D., Akbarnejad, A., Günnemann, S.: Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2847–2856 (2018)