Enhancing GNN-based Fraud Detectors against Camouflaged Fraudsters

CARE(Camouflage-resistant)-GNN

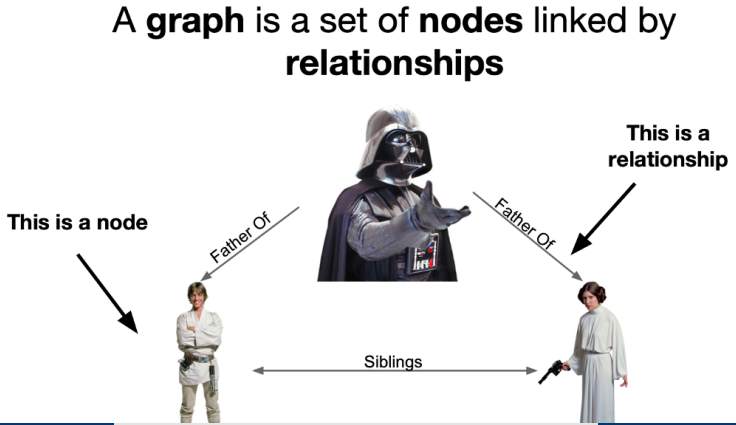
Jianping Bai

The Hong Kong University of Science and Technology (GZ)

October 31, 2025

Fraud Detection on Graphs: Why GNNs(Graph Neural Networks)?

- Entities (users, reviews, transactions) are linked by relations
- GNNs aggregate neighborhood information to better detect fraudsters.



Camouflage Types

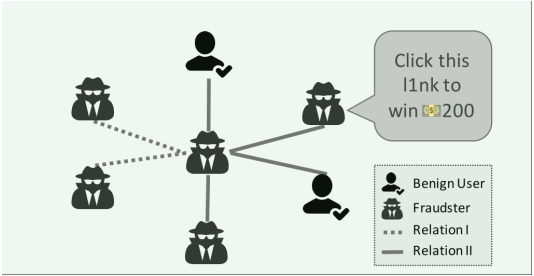
Drawbacks of traditional GNNs: **Camouflage** breaks naive neighbor aggregation.

Feature Camouflage

- Fraudsters mimic benign features (e.g., text tweaks, timing patterns).
- Makes feature-based or naive similarity unreliable.

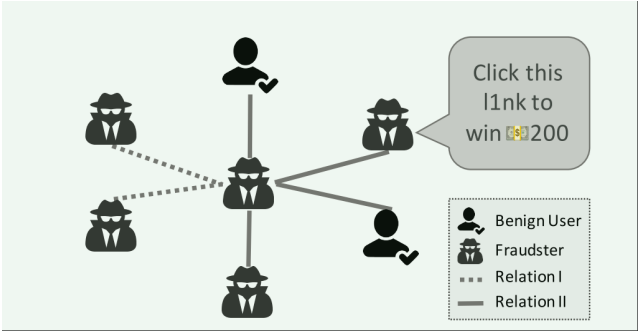
Relation Camouflage

- Fraudsters connect to many benign nodes under some relations.
- Noisy edges dilute suspiciousness during aggregation.



CARE-GNN: Three Modules

1. **Label-aware Similarity Measure** : measure the central node's and neighbors' similarity.
2. **Similarity-aware Neighbor Selector**: choose top-p most similar nodes under each relation; ratio p learned by RL.
3. **Relation-aware Aggregator**: aggregate nodes across different relations



Module 1: Label-aware Similarity

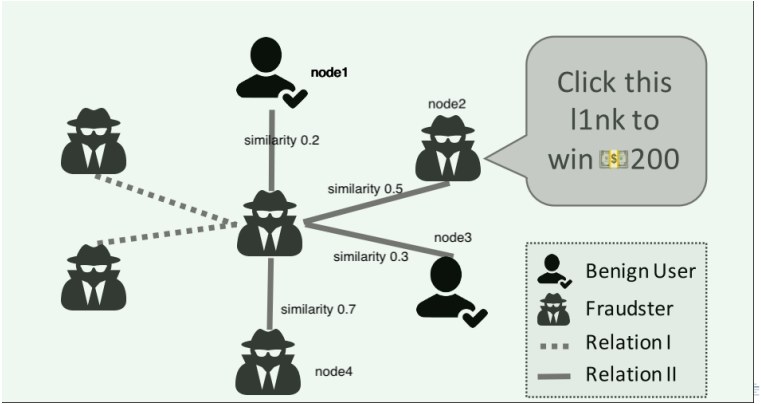
- **Traditional similarity method**: Only consider the feature's information. Fail to consider the **label's information**
- One-layer MLP predicts node label from $h_v^{(l-1)}$.
- Distance between v and v' at layer l:
  $D^{(l)}(v,v') = \left\| \sigma(\text{MLP}^{(l)}(h_v^{(l-1)})) - \sigma(\text{MLP}^{(l)}(h_{v'}^{(l-1)})) \right\|_1$.
- Similarity: $S^{(l)}(v,v') = 1 - D^{(l)}(v,v')$.

Some problem

- if the similarity measure could not effectively filter the camouflaged neighbors at the first layer,it will hamper the performance of following GNN layers.
- the MLP parameters can't be well-updated through back-propagation process.
- Loss: $\mathscr{L}_{\text{Simi}}^{(1)} = \sum_{v \in V} -\log\left(y_v \cdot \sigma(\text{MLP}^{(1)}(h_v^{(1)}))\right)$.

Module 2: Similarity-aware Neighbor Selector

- Similarity: $S^{(l)}(v, v') = 1 - D^{(l)}(v, v')$.Now we can calculate the similarity between any two nodes at layer l
- How do we choose **Top-p ratio** to exclude some neighbor's we do not want
- Rank result:node4(0.7),node2(0.5),node3(0.3),node1(0.2)



Module 2: Similarity-aware Neighbor Selector

- Need to adapt $p_r^{(l)} \in [0,1]$ automatically (varies across relations due to camouflage).
- **RL (Reinforcement Learning):**
  - **Action**: $a_r^{(l)} \in \{+\tau, -\tau\}$ to update ratio $p_r^{(l)}$ based on reward. original $p_r^{(l)}$ =0.5,updated $p_r^{(l)}$ = 0.5+0.02 or 0.5-0.02
  - **Reward** uses average neighbor distance change between previous epoch and current epoch

$$G\left(D_r^{(l)}\right)^{(e)} = \frac{\sum_{v \in V_{\text{train}}} D_r^{(l)}(v,v')^{(e)}}{|V_{\text{train}}|}. \tag{1}$$

$$f\left(p_r^{(l)}, a_r^{(l)}\right)^{(e)} = \begin{cases} +1, & G\left(D_r^{(l)}\right)^{(e-1)} - G\left(D_r^{(l)}\right)^{(e)} \geq 0, \\ -1, & G\left(D_r^{(l)}\right)^{(e-1)} - G\left(D_r^{(l)}\right)^{(e)} < 0. \end{cases} \tag{2}$$

Module 2: Similarity-aware Neighbor Selector

- **Terminal**:

$$\left| \sum_{e-10}^{e} f\left(p_r^{(l)}, a_r^{(l)}\right)^{(e)} \right| \leq 2, \quad \text{where } e \geq 10. \tag{3}$$

- This means RL converges in the recent ten epochs and indicates an optimal threshold $p_r^{(l)}$ is discovered

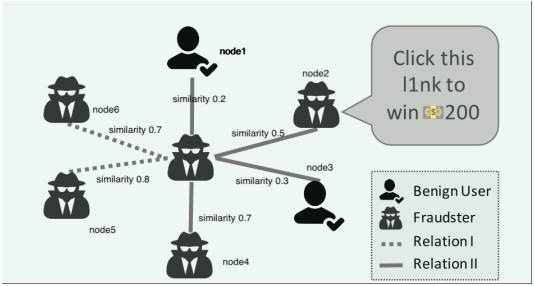
Only consider positive nodes for $p_r^{(l)}$ calculation

- accelerate the training process
- misclassifying a fraudster has a much higher cost to defenders than misclassifying a benign entity.
- a large number of benign entities already fuel sufficient information for the classifier

Module 3: Relation-aware Aggregator

Now we get the Top-p ratio for each relation $p_r^{(l)}$. So, for each relation, we can aggregate the information of every central node's neighbor.

Intra-relation aggregation (mean): $h_{v,r}^{(l)} = \text{ReLU}\Big(\text{AGG}_r^{(l)}(\{h_{v'}^{(l-1)} : (v, v') \in E_r^{(l)}\})\Big)$.



- Relation 1 $h_{v,1}^{(l)}$: p = 0.5,choose node4, node2 for aggregation(mean)
- Relation 2 $h_{v,2}^{(l)}$: p = 1,choose node5 and node 6 for aggregation(mean)

Module 3: Relation-aware Aggregator

Relation 1 $h_{v,1}^{(l)}$: p = 0.5,choose node4, node2 for aggregation(mean)

Relation 2 $h_{v,2}^{(l)}$: p = 1,choose node5 and node 6 for aggregation(mean)

- we could aggregate $h_{v,1}^{(l)}$ and $h_{v,2}^{(l)}$ together covering all relation,which calls Inter-relation combination
- **Inter-relation combination**: $h_v^{(l)} = \mathrm{ReLU}\Big(\mathrm{AGG}_{\mathrm{all}}^{(l)}(h_v^{(l-1)} \oplus h_{v,r}^{(l)}{}_{r=1}^{R})\Big)$.

How to choose aggregation method $\mathrm{AGG}^{(l)}$?
(mean, sum, attention weight, or other weight method?)

- Use $p_r^{(l)}$ as relation weights
- supposing we have selected the most similar neighbors under each relation, the attention coefficients or weighting parameters should be similar among different relations

Objective & Algorithms

- Final embedding $z_v = h_v^{(L)}$; classification loss: $\mathscr{L}_{\text{GNN}} = \sum_v -\log(y_v \cdot \sigma(\text{MLP}(z_v)))$.
- Joint loss: $\mathscr{L}_{\text{CARE}} = \mathscr{L}_{\text{GNN}} + \lambda_1 \mathscr{L}_{\text{Simi}}^{(1)} + \lambda_2 \|\Theta\|_2^2$.
- under-sampling to avoid overfitting(randomly sample the same number of negative instances as the number of positive instances).

Datasets & Graph Construction

Yelp (reviews as nodes)

- R-U-R: same user
- R-S-R: same product & star rating
- R-T-R: same product & month

Amazon (users as nodes)

- U-P-U: connect users reviewing at least one same product
- U-S-U: connects users having at least one same star rating within one week
- U-V-U: top 5% TF-IDF text similarity

Datasets & Graph Construction

Table 1: Statistics of Yelp and Amazon Datasets

Dataset	#Nodes (Fraud%)	Relation	#Edges	Feature Dim
Yelp	45,954 (14.5%)	*R-U-R*	49,315	32
		R-T-R	573,616	
		R-S-R	3,402,743	
		ALL	**3,846,979**	
Amazon	11,944 (9.5%)	*U-P-U*	175,608	25
		U-S-U	3,566,479	
		U-V-U	1,036,737	
		ALL	**4,398,392**	

Training Setup & Hyperparameters

- embedding size 64.
- Batch sizes: 1024 (Yelp), 256 (Amazon); optimizer Adam, lr=0.01.
- Under-sampling to balance positive/negative.
- RL step size $\tau = 0.02$; similarity loss weight $\lambda_1 = 2$; L2 weight $\lambda_2 = 0.001$.

Baselines: GCN, GAT, GraphSAGE, RGCN, GeniePath, GraphConsis, SemiGNN, Player2Vec + CARE variants.

Overall Performance

Table 3: Fraud detection performance (%) on two datasets under different percentage of training data.

	Metric	Train%	GCN	GAT	RGCN	Graph-SAGE	Genie-Path	Player-2Vec	Semi-GNN	Graph-Consis	CARE-*Att*	CARE-*Weight*	CARE-*Mean*	CARE-GNN
Yelp	AUC	5%	54.98	56.23	50.21	53.82	56.33	51.03	53.73	61.58	66.08	71.10	69.83	**71.26**
		10%	50.94	55.45	55.12	54.20	56.29	50.15	51.68	62.07	70.21	71.02	71.85	**73.31**
		20%	53.15	57.69	55.05	56.12	57.32	51.56	51.55	62.31	73.26	74.32	73.32	**74.45**
		40%	52.47	56.24	53.38	54.00	55.91	53.65	51.58	62.07	74.98	74.42	74.77	**75.70**
	Recall	5%	53.12	54.68	50.38	54.25	52.33	50.00	52.28	62.60	63.52	66.64	**68.09**	67.53
		10%	51.10	52.34	51.75	52.23	54.35	50.00	52.57	62.08	67.38	68.35	**68.92**	67.77
		20%	53.87	53.20	50.92	52.69	54.84	50.00	52.16	62.35	68.34	69.07	**69.48**	68.60
		40%	50.81	54.52	50.43	52.86	50.94	50.00	50.59	62.08	71.13	70.22	69.25	**71.92**
Amazon	AUC	5%	74.44	73.89	75.12	70.71	71.56	76.86	70.25	85.46	89.49	89.36	89.35	**89.54**
		10%	75.25	74.55	74.13	73.97	72.23	75.73	76.21	85.29	**89.58**	89.37	89.43	89.44
		20%	75.13	72.10	75.58	73.97	71.89	74.55	73.98	85.50	89.58	**89.68**	89.34	89.45
		40%	74.34	75.16	74.68	75.27	72.65	56.94	70.35	85.50	89.70	89.69	89.52	**89.73**
	Recall	5%	65.54	63.22	64.23	69.09	65.56	50.00	63.29	85.49	88.22	88.31	88.02	**88.34**
		10%	67.81	65.84	67.22	69.36	66.63	50.00	63.32	85.38	87.87	**88.36**	88.12	88.29
		20%	66.15	67.13	65.08	70.30	65.08	50.00	61.28	85.59	88.40	**88.60**	88.00	88.27
		40%	67.45	65.51	67.68	70.16	65.41	50.00	62.89	85.53	88.41	88.45	88.22	**88.48**

Figure 1: Overall performance

RL Dynamics(Yelp upper and Amazon lower)

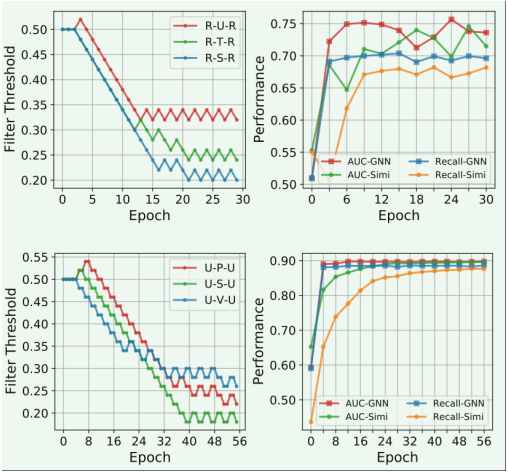


Figure 2: Threshold and performance

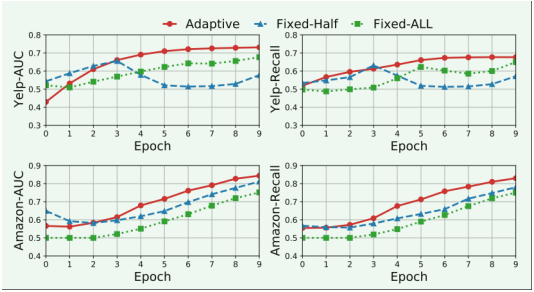


Figure 3: Different neighbor filtering methods

Hyper-parameter Sensitivity

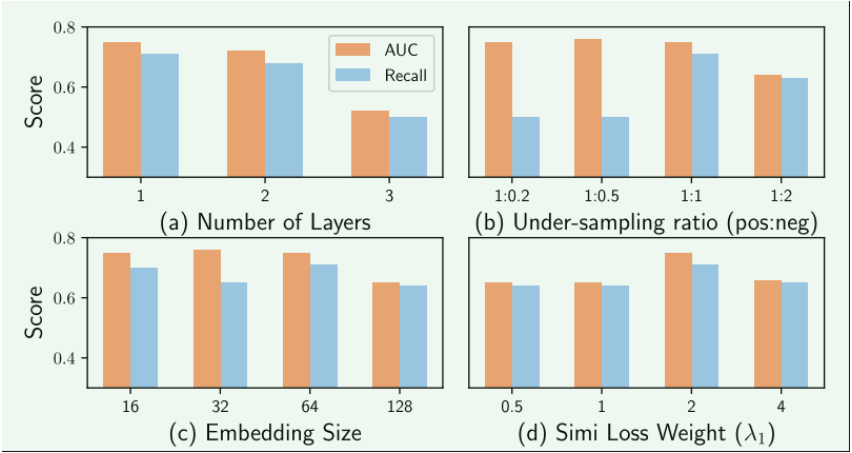


Figure 4: Parameter Sensitivity

Conclusion

- Dense-graph assumption (one-hop sufficient); sparse graphs may need deeper layers.
- CARE-GNN addresses *feature* and *relation* camouflage with three coordinated modules.
- RL-selected neighbor algos for each relation works.
- Strong empirical gains on Yelp & Amazon with efficient, simple design.

Thank you!