Deep Fraud Detection on Non-attributed Graph

Chen Wang*‡, Yingtong Dou*, Min Chen†, Jia Chen†, Zhiwei Liu* and Philip S. Yu*
*Department of Computer Science, University of Illinois at Chicago, USA
†GrabTaxi Holdings Pte Ltd, Singapore
{cwang266, ydou5, zliu213, psyu}@uic.edu, {min.chen, jia.chen}@grab.com

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9672028

***Abstract*—Fraud detection problems are usually formulated as a machine learning problem on a *graph*. Recently, Graph Neural Networks (GNNs) have shown solid performance on fraud detection. The successes of most previous methods heavily rely on rich node features and high-fidelity labels. However, labeled data is scarce in large-scale industrial problems, especially for fraud detection where new patterns emerge from time to time. Meanwhile, node features are also limited due to privacy and other constraints. In this paper, two improvements are proposed: 1) We design a graph transformation method capturing the structural information to facilitate GNNs on non-attributed fraud graphs. 2) We propose a novel graph pre-training strategy to leverage more unlabeled data via contrastive learning. Experiments on a large-scale industrial dataset demonstrate the effectiveness of the proposed framework for fraud detection.**

I. Introduction

The rapid growth of online services facilitates people's life while fosters fraudsters who reap monetary rewards and users' privacy via tampering with the system and policy. To name a few, the review spammer could boost the reputation of dishonest merchants in e-commerce and sabotage the recommender system [1]. Meanwhile, an increasing number of fraudsters have been engaging in social platforms and online financial services, according to a recent report [2].

To combat the fraudsters automatically, many machine learning approaches have been proposed [3]. As Graph Neural Networks (GNNs) achieving superior performance on many graph-related tasks [4], [5], many researchers and practitioners begin to adopt GNNs to detect fraud in various scenarios [6]–[9]. Most GNNs hold the graph homophily assumption where the connected nodes in a graph should have similar properties. Specifically, GNN recursively aggregates the node and its neighbor information to learn the node representation.

For the fraud detection problem, fraudsters usually behave insidiously, but their suspicious signals can be magnified when connecting them via shared entities like the IP address and the device [6]. For instance, if a group of coordinated fraudsters frequently use the same IP address and device, they would be closely connected on graphs built with the above entities. On the contrary, benign entities are more independent on the graph since they do not have coordinated behavior. After aggregating the neighbor information of the entities in the above graph using GNNs, fraudster entities' suspiciousness will become more significant compared to benign ones.

Despite the effectiveness of GNN-based fraud detectors, most of them depend on highly personalized graphs coupled with corresponding data which are not applicable to other problems [6], [9]. Meanwhile, the majority of previous works demand informative node features composed of user personal and behavioral information [7], [8]. However, the privacy and data retention policies may restrict the access of user information by companies in practice. Besides the challenge of constructing the graph, most real-world fraud detection tasks suffer from label scarcity issues since data annotation is labor-intensive due to the adversarial nature of fraudsters [8].

In this paper, we propose an approach that tackles two challenges above. **1)** For the challenge of graph construction in practice, we propose a generic graph structure and node feature initialization approach. We first introduce a graph transformation technique to convert commonly-used industrial graphs into smaller graphs while retaining useful information for downstream models. Then, inspired by recent work on node feature initialization for GNNs [10], [11], we leverage graph topological features to initialize node features for the non-attributed graph. **2)** To alleviate the label scarcity problem, we leverage the graph pre-training strategy, which is able to leverage more unlabeled data via graph contrastive learning [12]–[14]. Specifically, we devise a self-supervised GNN pre-training framework to capture the graph's topological properties across the unlabeled data. Then we generate inductive node embedding into the labeled dataset via the pre-trained graph encoder to train the fraud classifier.

Our framework has been validated by experimental results on a large-scale industrial dataset. The proposed graph construction approach and graph pre-training strategy can improve learning efficiency and boost fraud detection performance. Our contributions are summarized as follows:

- We propose a graph construction method for GNN-based fraud detection on the non-attributed graph.
- A graph pre-training strategy is adopted to alleviate the label scarcity problem in the fraud detection problem.
- Experimental results on a large-scale real world dataset validate various combinations of approaches.

II. Preliminary and Problem Definition

We take the loan-default detection task to demonstrate the widely-used graph prototype for fraud detection. As Fig. 1 (a) shows, a personal loan dataset usually includes entities like address, device, loan, and user. Most previous works build a *multi-entity graph* composed of different entities as nodes and their relations as edges (e.g., user-has-loan, user-uses-device) [6], [7], [9]. We can formally define the above

‡The work was done during Chen Wang's internship at Grab.

non-attributed multi-entity graph as $\mathcal{G}_m = (\mathcal{V}_m, \mathcal{E}_m, \mathcal{O}_\mathcal{V}, \mathcal{R}_\mathcal{E})$, where $v_i \in \mathcal{V}_m$ denotes the nodes, $\mathcal{E}_m$ denotes the edges. $\mathcal{O}_\mathcal{V}$ ($\mathcal{R}_\mathcal{E}$ resp.) represents the node types (relation types resp.).
Problem Definition. Given a non-attributed multi-entity graph, we first transfer it into an attributed *single-entity graph* $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s, \mathbf{X}^e, \mathbf{X}^v)$, where every $v_i \in \mathcal{V}_s$ belongs to the target entity to be classified. $\mathbf{X}^e$ and $\mathbf{X}^v$ represent the edge feature matrix and node feature matrix, respectively. With $\mathcal{G}_s$ and a set of partially annotated node labels $y_i \in Y, y_i \in (0, 1)$, where 0 (1 resp.) represents the *benign* (*suspicious* resp.) entity, we aim at training a classifier $f : \mathcal{G}_s \rightarrow Y$ to learn the representation of every $v_i \in \mathcal{V}_s$ and predict their labels.

III. Methodology

In this section, **1)** we first elucidate how we transform an attributed single-entity graph from a non-attributed multi-entity graph. Then, **2)** we present a graph pre-training strategy with contrastive learning to leverage the unlabeled data. Finally, **3)** we introduce how we encode the final node representation for fraud classification and how to fine-tune the GNN encoder.

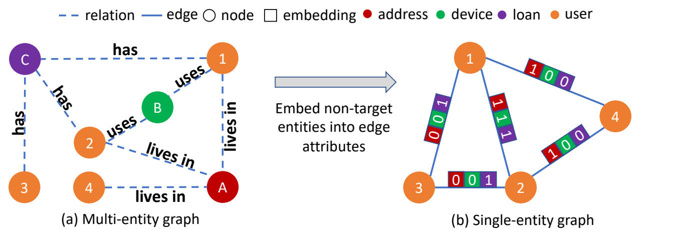

Fig. 1: Transferring a multi-entity graph into a single-entity graph.

A. Graph Construction

1) Structure transformation: As Fig. 1 (a) showing, many industrial graphs may contain entities like address and device, which only play the role to connect similar target entities (*i.e.*, users) while it is not easy to extract features from them. Since GNNs' capability and efficiency are largely dependent on node feature [10] and graph size [15], respectively, we propose to transfer all non-target entities as the feature of edges between target entities (as shown in Fig. 1). Specifically, the edge feature is a d-dimension one-hot vector, where d is the number of all non-target entity types, and each dimension indicates whether two end target entities are connected via the corresponding non-target entity.

The structure transformation above has three benefits: **1)** the single-entity graph shrinks the graph size significantly from the multi-entity graph via only keeping the target entities as nodes; **2)** the connections and their importance between target entities and different non-target entities are retained and can facilitate GNNs; **3)** for a center target node, a GNN could perceive more neighbor's information on the single-entity graph than the multi-entity graph with the same number of layers.

2) Node feature initialization: GNNs aim at learning node representations by learning the similarities shared between connected nodes. However, the expressive ability of a GNN is highly dependent on the quality of node features [10], [11]. Therefore, given the single-entity graph without target entity features, it would be better to initialize node features before feeding the graph into GNNs. Considering the costly feature engineering and the adversarial nature of fraudsters, we resort to adopting the following four graph topological features for an expeditious node feature initialization [10], [11].:

- *Random*: generating a feature vector for each node via sampling from a Gaussian distribution.
- *Degree* [5]: converting the node degree into a one-hot degree vector for each node, where the vector dimension depends on the maximum degree across all nodes.
- *PageRank* [16]: computing the PageRank score for each node, and use it as the node feature.
- *Eigen* [17]: applying the eigen decomposition on the normalized adjacency matrix of $\mathcal{G}_s$ and the top-k eigenvalues are the k-dimensional feature vector for each node.

We compare the performance of four nodes feature initialization approaches in Section IV-C.

B. Graph Pre-training

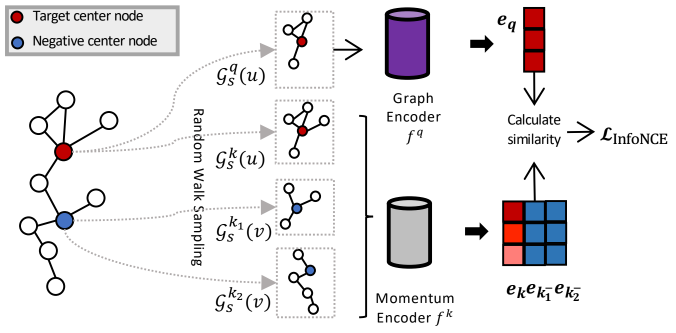

Fig. 2: Pre-training the GNN encoder on the unlabeled data.

As we mentioned in Section I, many real-world fraud detection tasks suffer from the label scarcity problem. Though GNNs can leverage unlabeled data during training, their capability is limited within the local neighborhood of target entities. Thanks to the recent advance in graph pre-training [18], [19], we adopt self-supervised contrastive learning to encode more information from unlabeled data. Specifically, under the assumption that common and transferable structural patterns exist across different nodes' sub-graphs, we can use plenty of unlabeled nodes to pre-train a GNN encoder to capture the local structure similarity between different nodes. Our graph pre-training framework has three steps: **1)** data augmentation, which constructs positive and negative sub-graph pairs of a node; **2)** GNN encoder, which maps one node's structural pattern to the latent representation; **3)** encoder training, which optimizes the GNN encoder with a contrastive loss.

1) Data augmentation: Contrastive learning requires one instance's positive pairs and negative samples to enhance the classifier's discriminative capability. Similar to previous work [19], we construct positive pairs as two sub-graphs of a node. Since two sub-graphs should share similar structure information to guarantee them to be positive pairs, we sample them from the r-ego network of a node.

We first conduct two iterations of random walk on node u's r-ego network $\mathcal{G}_s(u)$ (the superscript is ignored for simplicity), to generate two sub-graphs $\mathcal{G}_s^q(u)$ and $\mathcal{G}_s^k(u)$, which are regarded as a positive pair. After constructing positive sub-graph pairs for a node, we regard the sub-graphs generated from different nodes v's r-ego networks as the negative samples of the node. Fig. 2 demonstrates the sub-graph construction process, where $\mathcal{G}_s^q(u)$ and $\mathcal{G}_s^k(u)$ are a positive pair since they are sampled from the same node. $\mathcal{G}_s^{k_1}(v)$ and $\mathcal{G}_s^{k_2}(v)$ denote the negative samples of node u, which are sub-graphs sampled from the r-ego network of a different node.

2) GNN encoder: After retrieving positive and negative sub-graphs, we feed them into two graph encoders f^q and f^k, which are illustrated in Figure 2. We encode the sub-graph $\mathcal{G}_s(u^q)$ with graph encoder f^q, while encoding other sub-graphs with f^k. Correspondingly, we generate low-dimensional representation vectors $\mathbf{e}_q$ and $\mathbf{e}_k$ for the positive pair $\mathcal{G}_s(u^q)$ and $\mathcal{G}_s(u^k)$, respectively. We use the Graph Isomorphism Network (GIN) [20] as the GNN encoder to learn the node representation by only considering the node feature. This approach has two advantages: **1)** increasing the migration and generalization capabilities of the graph structure; **2)** decoupling the correlation between node features and edge features. The GNN encoder aggregation function is:

$$\mathbf{x}_i^{v\prime} = \text{MLP}\Big((1+\epsilon)\cdot\mathbf{x}_i^v + \sum_{j\in\mathcal{N}(i)}\mathbf{x}_j^v\Big), \tag{1}$$

where $\mathbf{x}_i^{v\prime}$ is v_i's embedding at the next GNN layer, ϵ is the weight parameter, $\mathbf{x}_i^v$ is v_i's embedding at the current layer, and $\mathbf{x}_j^v, j\in\mathcal{N}$ is the neighbor node embedding of v_i.

3) Encoder training: We adopt the contrastive loss named InfoNCE [21] to optimize the graph encoder in a self-supervised fashion, which maximizes the agreements between positive pairs. The InfoNCE loss is formulated as follows:

$$\mathcal{L}_{\text{InfoNCE}} = -\log\frac{\exp(\mathbf{e}_q^\intercal\mathbf{e}_k/\boldsymbol{\tau})}{\sum_{i=1}^n\exp(\mathbf{e}_q^\intercal\mathbf{e}_i/\boldsymbol{\tau})}, \tag{2}$$

where $\boldsymbol{\tau}$ is the temperature hyper-parameter. Minimizing Eq. (2) is equivalent to maximizing the similarity between positive pairs, *i.e.*, $\mathbf{e}_q$ and $\mathbf{e}_k$, while minimizing the similarity between negative pairs, *i.e.*, $\mathbf{e}_q$ and $\mathbf{e}_i$ where $i\neq k$. In practice, we view those instances as a query embedding $\mathbf{e}_q$ and a set of key embeddings $\{\mathbf{e}_i\}|_{i=1}^n$. The contrastive loss looks up a single key (denoted by $\mathbf{e}_k$) that $\mathbf{e}_q$ matches in the key set.

In contrastive learning, maintaining a K-size look-up key set is essential. Intuitively, as the denominator in Eq. (2) expresses, a larger key set size leads to better sampling of the underlying data space. To further improve the optimization process, we adopt the MoCo [22] training scheme, which maintains a dynamic set of keys with a queue and a moving-averaged encoder. MoCo is able to increase the key set size without additional back-propagation costs. Formally, if denoting the parameters of f_k as θ_k and those of f_q as θ_q, MoCo updates θ_k as $\theta_k \leftarrow m\theta_k + (1-m)\theta_q$, where $m\in[0,1)$ is a momentum hyper-parameter.

C. Model Fine-tuning

The pre-trained GNN encoder is then employed on the labeled graph to fine-tune embeddings. Specifically, we sample a sub-graph for each node and fine-tune the pre-trained encoder to encode the sub-graph. The objective for this fine-tuning step is to predict the associated label of each node. To note, in addition to node features, we aggregate both node and edge features of neighbors for final representation learning:

$$\mathbf{x}_i^{v\prime\prime} = \text{MLP}\Big((1+\epsilon)\cdot\mathbf{x}_i^v + \sum_{j\in\mathcal{N}(i)}\text{ReLU}(\mathbf{x}_j^v + \mathbf{x}_{ij}^e)\Big), \tag{3}$$

where $\mathbf{x}_{ij}^e$ is the edge features between node i and j. After passing the final node embedding to an MLP classifier, we adopt following cross-entropy loss to fine-tune the GNN encoder and the MLP parameters:

$$\mathcal{L}_{\text{CE}} = \sum_{v_i\in\mathcal{V}_{train}} -\log\left(y_i\cdot\text{ReLU}(\text{MLP}(\mathbf{x}_i^{v\prime\prime}))\right), \tag{4}$$

where $\mathbf{x}_i^{v\prime\prime}$ represents v_i's embedding after GNN's last layer.

IV. Experiment

A. Dataset

We evaluate our method on a large-scale industrial dataset. In Table I, by comparing the statistics of the multi-entity graph $\mathcal{G}_m$ and the single-entity graph $\mathcal{G}_s$, there is a large discrepancy between them. By transforming $\mathcal{G}_s$'s structure, the number of nodes has been reduced 26.69 times and edges by 22.07 times comparing with $\mathcal{G}_m$. Like many real-world fraud detection cases, the labeled data is very small. Only 0.188% of nodes are labeled on the graph used in this paper, which contains 1482 fraudulent and 1287 benign target entities. Utilizing the unlabeled data is critical for success, and our method has demonstrated the effectiveness of such a setting.

TABLE I: Dataset Statistics.

	# target entity	# non-target entity	$\lvert\mathcal{O}_\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$
$\mathcal{G}_m$	1,469,149	37,694,849	6	113,375,579
$\mathcal{G}_s$	1,469,149	0	1	5,136,750
$\mathcal{G}_m$:$\mathcal{G}_s$	26.69		6	22.07

B. Experimental Settings

We compare four feature initialization methods proposed in Section III-A2 under various settings. We adopt a 3-layer GNN encoder for all settings. For pre-training, batch size is 200, embedding dimension is 16, and learning rate is $1e^{-6}$. For fine-tuning, batch size is 100, embedding dimension is 32, and learning rate is $1e^{-5}$. We conduct five-fold experiments and report the average *micro-F1 score* in Table II.

C. Result Analysis

Besides the node embedding, we also use the 3-hop sub-graph embedding for fraud classfication. Table II shows the experimental results, and we have the following findings:

TABLE II: Fraud detection performance (micro-F1 score), "PT" represents using pre-training, "NE" and "SE" represent node embedding and sub-graph embedding, respectively.

Method	**multi-entity graph**		**single-entity graph**	
	NE	SE	NE	SE
Random	0.436	0.439	0.441	0.447
Random + PT	0.424	0.427	0.405	0.410
Degree	0.472	0.481	0.488	0.479
Degree + PT	0.499	0.483	0.532	0.541
PageRank	0.566	0.559	0.594	0.610
PageRank + PT	0.641	0.633	0.661	0.674
Eigen	0.630	0.628	0.679	0.683
Eigen + PT	0.623	0.644	0.708	**0.721**

1) The single-entity graph is better than the multi-entity graph: The best classification result in the single-entity graph exceeds that of the multi-entity graph by 11.95%. For center node, a GNN can reach farther target entities on the single-entity graph compared to that of multi-entity graphs, thus more informative nodes are encoded on the single-entity graph.

2) Graph pre-training is helpful: We can observe a performance boost for most feature initialization methods after applying the proposed graph pre-training strategy (see method + PT in Table II). This result indicates the effectiveness of using the contrastive learning to pre-train the GNN encoder on the unlabeled data, and it boosts 5.56% micro-F1 score compare Eigen + PT with Eigen. On the contrary, graph pre-training has a negative effect on randomly initialized nod features. A possible reason is that the random feature carries no topological information of the node and thus can not be leveraged by self-supervised contrastive learning.

3) Sub-graph embedding is more effective than node embedding: As shown in Table II, using sub-graph embedding in the single-entity graph has noticeable improvement while the multi-entity graph is not. It suggests that the single-entity graph has a better representation than the multi-entity graph with the same hops, and insufficient neighbor information will bring more harm rather than gain.

D. Embedding Visualization

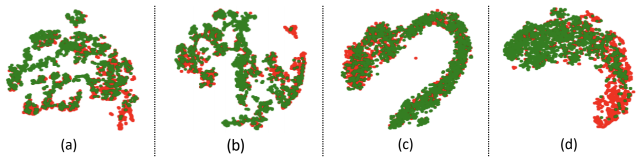

Fig. 3: t-SNE plots of node embeddings obtained by: (a) Eigen + multi-entity; (b) Eigen + single-entity; (c) Eigen + PT + multi-entity; (d) Eigen + PT + single-entity. Red: fraudster, green: benign entity.

To understand the difference between the proposed methods straightforwardly, we adopt t-SNE [23] to visualize the node and sub-graph embeddings generated by the GNN encoders in Fig. 3. It can be seen from the figure that graph pre-training not only gathers similar nodes together but also increases the discriminative capability of the GNN encoder.

V. Conclusion and Future Work

In this paper, we propose a framework to transform a general non-attributed graph in industry and initialize its node and edge features for GNN-based fraud detection. The experimental results on a large-scale industrial dataset demonstrate the effectiveness of the proposed framework. Future work includes investigating more informative node feature initialization approaches and optimize the graph pre-training efficiency.

Acknowledgments

This work is supported in part by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941.

References

[1] S. Zhang, H. Yin, T. Chen, Q. V. N. Hung, Z. Huang, and L. Cui, "Gcn-based user representation learning for unifying robust recommendation and fraudster detection," in *SIGIR*, 2020.
[2] Datavisor, "Digital fraud trends report 2021," 2021. [Online]. Available: https://bit.ly/3BAQidA
[3] M. Jiang, P. Cui, and C. Faloutsos, "Suspicious behavior detection: Current trends and future directions," *IEEE Intelligent Systems*, 2016.
[4] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017.
[6] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous graph neural networks for malicious account detection," in *CIKM*, 2018.
[7] A. Li, Z. Qin, R. Liu, Y. Yang, and D. Li, "Spam review detection with graph convolutional networks," in *CIKM*, 2019.
[8] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *CIKM*, 2020.
[9] C. Liu, L. Sun, X. Ao, J. Feng, Q. He, and H. Yang, "Intention-aware heterogeneous graph attention networks for fraud transactions detection," in *KDD*, 2021.
[10] C. T. Duong, T. D. Hoang, H. T. H. Dang, Q. V. H. Nguyen, and K. Aberer, "On node features for graph neural networks," *arXiv preprint arXiv:1911.08795*, 2019.
[11] H. Cui, Z. Lu, P. Li, and C. Yang, "On positional and structural node features for graph neural networks on non-attributed graphs," *arXiv preprint arXiv:2107.01495*, 2021.
[12] N. Huyan, D. Quan, X. Zhang, X. Liang, J. Chanussot, and L. Jiao, "Unsupervised outlier detection using memory and contrastive learning," *arXiv preprint arXiv:2107.12642*, 2021.
[13] J. Tack, S. Mo, J. Jeong, and J. Shin, "Csi: Novelty detection via contrastive learning on distributionally shifted instances," *arXiv preprint arXiv:2007.08176*, 2020.
[14] B. Chen, J. Zhang, X. Zhang, Y. Dong, J. Song, P. Zhang, K. Xu, E. Kharlamov, and J. Tang, "Gccad: Graph contrastive coding for anomaly detection," *arXiv preprint arXiv:2108.07516*, 2021.
[15] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graph-saint: Graph sampling based inductive learning method," *ICLR*, 2020.
[16] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, 1998.
[17] Q. Huang, H. He, A. Singh, S.-N. Lim, and A. R. Benson, "Combining label propagation and simple models out-performs graph neural networks," *arXiv preprint arXiv:2010.13993*, 2020.
[18] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," in *KDD*, 2020.
[19] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *NeurIPS*, 2020.
[20] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.
[21] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.
[22] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020.
[23] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci, "Visualizing high-dimensional data: Advances in the past decade," *IEEE TVCG*, 2016.