Heterogeneous Mini-Graph Neural Network and Its Application to Fraud Invitation Detection

Yong-Nan Zhu*, Xiaotian Luo†, Yu-Feng Li*
Bin Bu†, Kaibo Zhou†, Wenbin Zhang†, Mingfan Lu†
Nanjing University*, iQIYI Inc†
{zhuyn,liyf}@lamda.nju.edu.cn,{luoxiaotian,bubin,zhoukaibo01,zhangwenbin,lumingfan}@qiyi.com

ABSTRACT

Effectively detecting the fraudulent invitations is valuable for many online Internet enterprises such as iQIYI to promote good products and improve the user experience. However, it remains highly non-trivial to address, which mainly lies in two challenging data characteristics. First, the invitation graph structure is globally large yet locally small, as a large number of invitations usually occur in a very small local graph, making the global and local consistency difficult to achieve simultaneously. Secondly, the user associations are heterogeneous and diverse, as the user associations are from multiple different data resources, making the effects of multiple user associations difficult to use effectively. To this end, this paper proposes a novel heterogeneous graph neural network HmGnn, to detect fraudulent invitations at iQIYI platform. To the best of our knowledge, this is the first attempt to study fraud invitation detection via graph neural networks. HmGnn handles the homogeneity and heterogeneity of networks simultaneously. Specifically, the proposal constructively introduces links between homogenous mini-graphs based on the similarity of mini-graphs, facilitating the impact of local mini-graphs to the global graph structure. In addition, this paper presents a heterogeneous attention convolution network to accurately optimize the contribution of multiple heterogeneous user associations. Extensive experiments conducted on real-world business data validate the excellent effectiveness and improvement on risk management of our method.

KEYWORDS

Fraud Invitation Detection, Graph Structure, Mini-Graph, Heterogeneous Graphs, Graph Neural Network

1 INTRODUCTION

Data with relationships is often represented in the form of graphs, such as social networks, economic networks and chemical molecules. Besides, a mass of users are active on large-scale graph-structure-based Internet platforms and applications, who often contain rich yet diverse relationships. In particular, nowadays, Internet companies often offer incentives, such as monetary rewards/discount coupons/membership cards, encouraging regular subscribers to invite new users to promote the products. The invitation relationship consequently forms a natural link between inviters and invitees. However, due to the lucrative rewards of successful invitations, there are often fraudulent users to invite fake people that never carry out any meaningful activity after the invitation procedure is finished. Such inviters and invitees pursue excess profits, and may even cash out for ill-gotten gains, which are inherently harmful to the platform's ecosystem. Effectively detecting the fraudulent users is valuable, for many online Internet enterprises such as iQIYI to promote good products and improve the user experience.

Traditional methods based on analyzing abnormal behavior and detecting suspicious patterns are mainly spread in three directions. First, rule-based methods [18] directly generate complicated rules which requires expensive human resources and lacks generalizability. Second, social connectivity based approaches reformulate the problem by considering the connectivities among accounts [8, 15]. This approach requires domain knowledge of the whole social graph, which is very hard to generalize for many realistic scenarios. Third, machine learning based classifiers learn statistic models by exploiting a large amount of historical data to infer fake users [1, 5]. Feature selection plays a critical role for most of these methods, which would be annoying and cumbersome in practice.

However, none of the above methods are appropriate to well address the difficulties occurred in fraud invitation detection. On one hand, the invitation graph structure is globally large yet locally small, as a large number of user invitation relationships usually occur in a very small local part. We call this kind of small graph structure as *mini-graph*. That is to say, the whole invitation graph has many disconnected components and each component is connected and defined as *mini-graph*. Besides, mini-graphs are in a long-tailed distribution, as illustrated in the first and third subgraphs of Figure 1. On the other hand, the user associations are heterogeneous and diverse, as the user association comes from various types of information resources, such as IP address, device ID, etc, as illustrated in the second subgraph of Figure 1. Such data characteristic makes the effects of multiple user associations difficult to explore effectively and lightly.

To tackle above situations, a novel graph neural network method HmGnn (**H**eterogeneous **M**ini-**G**raphs **N**eural **N**etwork) is proposed, which is verified on the iQIYI platform, an innovative market-leading online entertainment service. HmGnn first constructively introduces links between mini-graphs based on the similarity of mini-graphs. Along with this strategy, the structural information about different mini-graphs is shared, which increases the correlation between mini-graphs and facilitates the impact of mini-graphs

DLG-KDD'20, August 24, 2020, San Diego, California USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

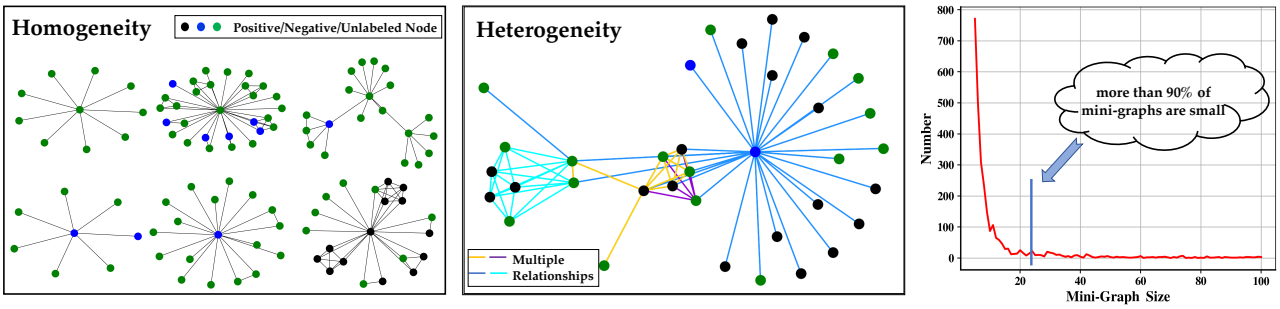


Figure 1: Illustration of the situations in fraud invitation detection based on the real data at iQIYI platform. The left figure shows the homogeneity of mini-graphs which are typically small. A connected component of 36 users linked via heterogeneous associations is visualized in the middle figure. Different colors of nodes and edges indicate different class labels and relationships. The distribution of mini-graph size plotted in the right figure shows it meets a long-tailed distribution.

to the global graph structure. To effectively exploit the heterogeneity of user associations, we present an attention mechanism to accurately optimize the contributions of multiple associations adaptively, where the attention coefficients reveal the reliability in identifying fraudulent users. Extensive experiments on the iQIYI platform are conducted to evaluate the superiority of the proposed model compared with state-of-the-art supervised baselines and GNNs. The results demonstrate that the new method consistently achieves performance gain under various criteria and effectively reduces the actual economic loss caused by fraudulent invitations.

2 BACKGROUND

Graph neural networks (GNNs) [6] aim to extend the deep neural networks to deal with non-Euclidean data with complex relationships and interdependency. In particular, there are two types of GNNs that are highly related to the work of this paper. One is Graph Convolutional Network (GCN) [10], which is good at handling graph structured data with a small number of labels; the other is heterogeneous GNNs [2, 17], which is used to handle heterogeneous graph structures. Specifically, GCN constructs node embeddings by mixing the embeddings of neighbors which is a localized first-order approximation of spectral graph convolutions. The $(l+1)$-th layer embedding is defined as:

$$\mathbf{H}^{l+1} = \sigma\left(\tilde{\mathbf{A}}\mathbf{H}^{l}\mathbf{W}^{l}\right) \tag{1}$$

where $\tilde{\mathbf{A}}$ is a symmetric normalization of adjacency matrix $\mathbf{A}$ with self-loops, i.e., $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$, $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\hat{\mathbf{D}}$ is the diagonal node degree matrix of $\hat{\mathbf{A}}$, $\mathbf{W}^{l}$ is a free parameter and $\sigma(\cdot)$ is the activation function. Despite its simpleness, GCN achieved state-of-the-art results on various tasks. However, previous study [19] has shown that GCNs are incapable of distinguishing different types of data associations. Treating different types of relationships as the same would be sub-optimal or even harmful.

Heterogeneous graphs examine scenarios with nodes and/or edges of various types. GATNE [2] considers self-attention mechanism to aggregate heterogeneous neighborhood information to generate node embeddings in the recommendation system. GEM [13] adaptively learns discriminative embeddings from heterogeneous account-device graphs for malicious account detection. DRL [14] proposes a deep reinforcement learning based approach to study the problem of curriculum learning for node representation in heterogeneous star networks. As can be seen that the construction of heterogeneous GNNs is oriented to a certain graph structure, e.g., neighborhood, account-device graph, star network, to design the representation fusion mechanism. To the best of our knowledge, the exploration of designing a heterogeneous fusion mechanism for the graph structure such as mini-graph has not been studied.

3 METHODOLOGY

In this section, we formally present HmGnn which first generates hyper-graphs and optimizes the generation by their similarities. A heterogeneous graph network is adopted to utilize multiple relationships and finally we present model training with analysis.

3.1 Hyper-Graph Generation

It is well known that users interact with others when they surf the Internet, such as inviting friends to use the same app or playing together on the same online platform. Due to the limitation of the user's social ability, however, the graphs are usually very small, as shown in the right figure of Figure 1. Existing study [19] shows that common GNNs are incapable of distinguishing different graph structures, not to say sharing label and feature information among naturally separated mini-graphs.

To solve this challenge, we need to address two sub-tasks: the representation of the connected mini-graphs and the generation of their links. For the former, we introduce hyper-nodes to represent mini-graphs and use the associations of hyper-nodes to tackle the latter. For simplicity and distinction, we call the nodes in originally connected graphs as normal nodes and the generated graph as hyper-graph which contains normal nodes and hyper-nodes. Specifically, for a mini-graph $G_o = (V_o, E_o)$, its representation (hyper-node) is generated by the mean feature of all its internal normal nodes and fixed during the training process:

$$\mathbf{x}_{G_o} = \frac{1}{|V_o|}\sum_{v \in V_o}\mathbf{x}_v \tag{2}$$

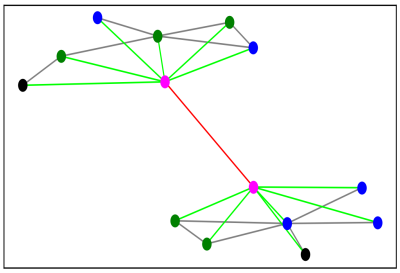


Figure 2: Hyper-graph generation. There are three types of hyper-graph edges, namely, between hyper-nodes (in red), between normal nodes (in gray), between hyper- and normal node (in green).

To this end, there are three types of edges, as shown in Figure 2. Normal nodes are connected to the hyper-node generated by its mini-graph, to ensure the fair and unbiased transmission of information. The purpose of the link between hyper-nodes and hyper-nodes is to share the structural information of the mini-graphs, thus realizing the label propagation from local to global. The following theorem provides theoretical support and shows that even if the hyper-nodes are connected randomly and the number of edges of the hyper-nodes is only a logarithm level of hyper-nodes, the connected components can be effectively reduced.

Theorem 1. *Suppose there are h hyper-nodes each of which is connected to exactly k other randomly chosen hyper-nodes to build graph G_h. Graph G_h is connected if there exists at least one path between any two nodess in G_h. If $k = \lambda \frac{\ln h}{2}$ where $\lambda > 1$, for large h,*

$$\mathbb{P}(G_h \text{ is disconnected}) \to 0$$

Proof. Due to the randomness and independence of constructing edges in the graph, for large h, the probability that any two points have edges is $p = \frac{2k}{h-1} = \frac{\lambda \ln h}{h-1}$ which is a special instance of Erdós-Rényi Graph [4]. Let b_i be a Bernoulli random variable defined as:

$$b_i = \begin{cases} 1 & \text{if node } i \text{ is isolated,} \\ 0 & \text{otherwise.} \end{cases}$$

The probability of an individual node being isolated is:

$$\mathbb{P}(b_i = 1) = (1-p)^{h-1} \tag{3}$$

The event "graph is disconnected" is equivalent to the existence of i nodes without an edge to the remaining nodes, for some $i \le h/2$. Let A_i denote the event that there are i nodes that are not connected to the rest. The likelihood of G_h being disconnected is determined by union bound of all events A_i

$$\mathbb{P}(G_h \text{ is disconnected}) \le \sum_{i=1}^{h/2} \mathbb{P}(\mathrm{A_i \text{ happens}}) = \sum_{\mathrm{i=1}}^{\mathrm{h/2}} \binom{\mathrm{h}}{\mathrm{i}} (1-\mathrm{p})^{\mathrm{i(h-1)}}$$

Using Stirling's formula $x! \sim \left(\frac{x}{e}\right)^x = \exp(x \ln x - x)$ and the convergence $(1 + \frac{1}{x})^x \approx e$ to approximate two multipliers, we have

$$\begin{aligned} \mathbb{P}(G_h \text{ is disconnected}) &\le \sum_{i=1}^{h/2} \exp\left(h f_h(\frac{i}{h})\right) \\ &\approx C_o \exp\left(h \max_{\frac{1}{h} \le x \le \frac{1}{2}} f_h(x)\right) \end{aligned} \tag{4}$$

where C_o is a constant that is independent of h, $f_h(x) = -x \ln x - (1-x)\ln(1-x) - \lambda x(1-x)\ln h$. It can be checked that the maximum over $[\frac{1}{h}, \frac{1}{2}]$ of $f_h(x)$ is achieved at $x^\star = \frac{1}{h}$ and $f_h(x^\star) \approx (1-\lambda)\frac{\ln h}{h}$

$$\mathbb{P}(G_h \text{ is disconnected}) \le C_o h^{1-\lambda} \xrightarrow{\lambda > 1} 0$$

□

Theorem 1 shows that after the construction of connecting each hyper-node to $k = o(\ln h)$ other randomly chosen hyper-nodes, the generated graph is connected in a large probability, which is conducive to share information between naturally separated mini-graphs. According to specific graph structure, we can also use the value of k to control the degree of connection. Because random connection will introduce many irrelevant edges, we can further improve the construction of graph by constructing the edges between hype-node through k-nearest neighbors via the feature representation as shown in Eq. (2). Experiments show that such an approach is effective, and it is not difficult to determine the hyperparameter k.

3.2 Heterogeneous Graph Convolution

We firstly separate the graph $G = (V, E)$ into $|\mathcal{D}|$ subgraphs: $\{G^d = (V, E^d) \mid d \in \mathcal{D}\}$ each of which preserves all the vertices in G, but only the dth-type edges. As a result, there are $|\mathcal{D}|$ adjacency matrices $\{A^d \mid d \in \mathcal{D}\}$ and different types of edges have different influences for hidden representation. Similar to the work in [13], the attention mechanism and residual style connection [7] that embeds vanilla feature into the hidden state are introduced to adaptively estimate the importance of different relationships. Before taking the heterogeneous graph convolution, as introduced in Section 3.1, we construct the hyper-graphs $G_h^d = \{V_h, E_h^d\}$ where $V_h(E_h^d)$ includes normal nodes(edges) set $V(E^d)$ and hyper-nodes(edges). Then the l^{th} layer convolution is defined as:

$$\mathbf{H}^{l+1} = \sigma\left(\mathbf{X}_h \mathbf{U}^l + \sum_{d \in \mathcal{D}} att_d \left(\tilde{\mathbf{A}}_h^d \mathbf{H}^l \mathbf{W}_d^l + \mathbf{b}_d^l\right)\right) \tag{5}$$

where $att_d = \mathrm{softmax}(\alpha_d) = \exp(\alpha_d)/\mathcal{Z}$ is the attention coefficient for graph G_h^d with $\mathcal{Z} = \sum_i \exp(\alpha_i)$, $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_{|\mathcal{D}|}]$, $\mathbf{U}^l$, $\mathbf{W}_d^l$ and $\mathbf{b}_d^l$ are free parameters that need to be estimated. The above vanilla feature embedding at each layer is shared among different subgraphs which is beneficial to reduce computational complexity and the risk of overfitting.

Specifically, $\mathbf{H}^0 = \mathbf{X}_h = [\mathbf{X}; \mathbf{X}_h^1; \cdots; \mathbf{X}_h^{|\mathcal{D}|}]$ is the universal representation of the whole graph which concatenates the feature matrix of normal- and hyper- nodes. In order to aggregate the representation of one node itself and its neighbors based on $\mathbf{X}_h$, we

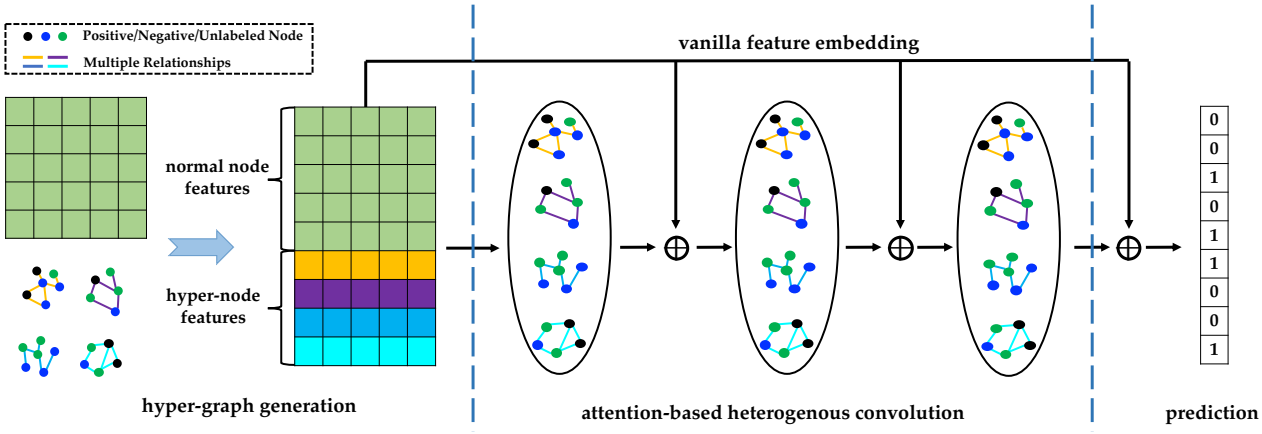


Figure 3: The overall architecture of our proposal. The left part illustrates hyper-graph generation given normal node features and their natural links. We concatenate the feature matrix of the generated hyper-nodes from different subgraphs to the input feature matrix to form the final feature representation and it is fed into the middle procedure to obtain the prediction. Different colors of nodes and edges indicate different types of nodes and relationships.

use *reindex* trick to construct the adjacency matrix of graph G_h^d:

$$\mathbf{A}_h^d = \begin{bmatrix} \mathbf{A}^d & \cdots & \mathbf{A}_{nh}^d & \cdots \\ \vdots & \ddots & \vdots & \ddots \\ \mathbf{A}_{hn}^d & \cdots & \mathbf{A}_{hh}^d & \cdots \\ \vdots & \ddots & \vdots & \ddots \end{bmatrix} \in \mathbb{R}^{(n+n')\times(n+n')}$$

where $n' = \sum_d h_d$ is the total number of hyper-nodes, $\mathbf{A}^d \in \mathbb{R}^{n\times n}$, $\mathbf{A}_{nh}^d = (\mathbf{A}_{hn}^d)^{\mathrm{T}} \in \mathbb{R}^{n\times h_d}$ and $\mathbf{A}_{hh}^d \in \mathbb{R}^{h_d\times h_d}$ are the adjacency matrix of normal-normal, normal-hyper and hyper-hyper nodes, respectively, as illustrated in Fig. 2. For other elements in $\mathbf{A}_h^d$, they are all zero values which means that there is an offset in representing the hyper-nodes from different subgraphs. Finally, $\tilde{\mathbf{A}}_h^d$ is obtained from $\mathbf{A}_h^d$ by the renormalization transformation [10] to make the learning process more steady.

3.3 Model Training and Analysis

For the semi-supervised node classification task, we stack multiple heterogeneous graph convolution layers as defined in Eq. (5) with a softmax(·) activation function on the output of the last layer to obtain the final inferred labels $\tilde{Y}$. We minimize the cross-entropy loss over all labeled examples and model complexity simultaneously.

$$\min_{\boldsymbol{W},\boldsymbol{U},\boldsymbol{\alpha},\mathbf{b}} \mathcal{L}(\boldsymbol{W},\boldsymbol{U},\boldsymbol{\alpha},\mathbf{b}) = -\frac{1}{|\mathbf{V}_L|}\sum_{v\in\mathbf{V}_L}\sum_{c=1}^{C} Y_v^c \ln(\tilde{Y}_v^c) + \beta\|\boldsymbol{\Theta}\|_2 \quad (6)$$

where $\mathbf{V}_L$ is the labeled node set, $\boldsymbol{\Theta} = \{\boldsymbol{W},\boldsymbol{U},\boldsymbol{\alpha},\mathbf{b}\}$ indicates the parameters, C is the number of distinct classes and β is a hyperparameter to trade-off the generalization and complexity of the model. A schematic description of our model is given in Fig. 3.

Suppose that there are n nodes and m edges in the graph. Since the hyper-graph generation is conducted on each of $|\mathcal{D}|$ subgraphs, this operation will generate additional $n' = \sum_{d=1}^{|\mathcal{D}|} h_d$ nodes and $m' = |\mathcal{D}|n + \sum_{d=1}^{|\mathcal{D}|} kh_d$ edges respectively where h_d is the number of generated hyper-nodes in the d^{th} subgraph. The time and memory complexity are $O(T(m+m')dim + |\mathcal{D}|T(n+n')dim^2)$ and $O(|\mathcal{D}|T(n+n')dim + Tdim^2)$ where T is the number of layers and the dimensions of hidden representation denoted by dim.

4 EXPERIMENTS

In this section, we report the comparison results of HmGnn[1] with several competitors on the real data of iQIYI platform [2] and give some discussions and conclusions. We also verify the effectiveness of several sub-routines and study parameter sensitivity, time cost. Finally, we analyze the contribution of different types of edge and its practical intuition and give the conclusion of the online test.

4.1 Dataset

A two-month data from some experimental businesses of iQIYI platform in 2019 from April to June is used to evaluate various machine learning models. The connection between users can be invitations and shared devices, such as IP addresses and physical devices. We use two device identifiers (DID1 and DID2) to identify the same physical devices. There are 4 kinds of user associations in total. If two users share the same device (IP, DID1, DID2), one edge with weight one is built. For invitation relationship, the weights of edges between inviters and invitees are equal to the invitation frequency. All edges are undirected for universality and multiple edges between two users may exist due to multiple associations. The whole data includes 82,767 nodes and 152,818 edges, of which 17,346 are labeled samples and 8,446 are fraudulent users (positive samples). In addition, 8,900 normal users are selected as negative samples[3]. Note that the business data is preprocessed anonymously and cryptographically to protect user privacy.

[1] The source code is available at https://github.com/iqiyi/HMGNN.
[2] https://www.iqiyi.com/
[3] In Internet companies such as iQIYI, different users have different VIP levels. In the experiment, the users with higher level are selected as the normal samples due to their long participation time and high participation frequency.

Table 1: Quantitative results (mean ± std) of our proposal and competing methods. The best results are bolded.

Methods	Precision	Recall	F1	AUC
LR	0.8681 ± 0.0078	0.8120 ± 0.0108	0.8390 ± 0.0070	0.9123 ± 0.0044
XGBoost	0.8987 ± 0.0057	0.8182 ± 0.0084	0.8566 ± 0.0049	0.9296 ± 0.0023
MLP	0.8687 ± 0.0105	0.8572 ± 0.0083	0.8628 ± 0.0031	0.9312 ± 0.0026
GCN	0.8054 ± 0.0296	0.8649 ± 0.0266	0.8332 ± 0.0055	0.9033 ± 0.0067
GAT	0.8828 ± 0.0128	0.8096 ± 0.0218	0.8443 ± 0.0092	0.9114 ± 0.0054
ASGCN	0.8705 ± 0.0099	0.8418 ± 0.0101	0.8558 ± 0.0027	0.9040 ± 0.0021
mGCN	0.8948 ± 0.0107	0.8639 ± 0.0146	0.8789 ± 0.0059	0.9435 ± 0.0023
HmGnn	**0.9058 ± 0.0079**	**0.8844 ± 0.0069**	**0.8949 ± 0.0031**	**0.9554 ± 0.0015**

4.2 Experimental Setting

4.2.1 Competitors. We consider several representative methods for the classification task, including both supervised baselines and state-of-the-art graph neural networks: Logistic Regression (LR), XGBoost [3], Multi-Layer Perceptron (MLP), GCN[4] [10], GAT[5] [16], ASGCN[6] [9] and mGCN which is a modified version of GCN [10]. For mGCN, we separate the graph into four different graphs by edge type and it propagates information according to Eq. (5) without generating hyper-nodes and introducing attention mechanism and residual connection.

For supervised methods, including LR, XGBoost, and MLP, we use both training and validation datasets to learn the classifier. For GNNs, including GCN, GAT, ASGCN, mGCN and our proposal, the training dataset is used to learn the model while the model selection is based on the performance on the validation dataset. The final performance is tested on the testing dataset for all methods. For XGBoost, the number of estimators is 100 and the learning rate is 0.01. For GNNs and MLP, we stack three layers of neural network with the same embedding size which is 32 except for the first layer whose size is 128. We use 16 heads per hidden layer in GAT to make a trade-off between efficiency and effectivity. The ratio of variance reduction in ASGCN is set to 0.5. For our proposal, k and β are set to 10 and 0.0005, respectively. We conduct 10 independent and random runs of all methods on a 32-core Intel Xeon CPU (3.20 GHz) server with 64GB RAM and report their average performance.

4.2.2 Evaluation Metrics. We mainly consider two measures: F1 and AUC. which play a key role in evaluating different models in risk management for Internet companies. We also report Precision and Recall to make a more comprehensive presentation.

4.3 Performance Analysis

4.3.1 Comparison Results. Table 1 shows the comparison results. As for why the phenomenon that supervised models outperform traditional semi-supervised GNNs happens, the heterogeneous graph structure introduces great difficulties to learn a better hidden representation from both vanilla features and structure input with only roughly one in eight supervised information. However, if the heterogeneity is taken advantage of in a more appropriate way to learn a model, more promising results are obtained while comparing GCN with mGCN. This finding emphasizes the difficulty and importance of utilizing the heterogeneous graph simultaneously. Besides, our proposed method still achieves the best performance considering all evaluation metrics which shows the potential of robustness and effectivity of our approach. Overall, HmGnn achieves performance improvement over the best baseline by 0.71%, 1.95%, 1.6% and 1.19% on the four metrics, respectively.

[4]https://github.com/tkipf/gcn
[5]https://github.com/PetarV-/GAT
[6]https://github.com/huangwb/AS-GCN

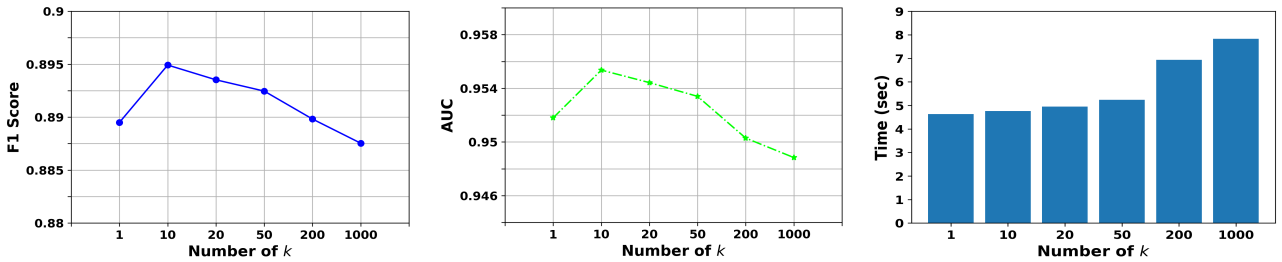
Table 2: Quantitative results (%) of ablation study. The abbreviation *"wo"* stands for *"without"*. The values in the brackets indicate the ratio of performance degeneration.

Metrics	wo_hyper	wo_residual	wo_attention
F1	89.00 (-0.49)	88.90 (-0.59)	89.15 (-0.34)
AUC	95.24 (-0.30)	95.08 (-0.46)	95.17 (-0.37)

4.3.2 Ablation Study. HmGnn is a joint learning framework for diverse heterogeneous graphs with sparse and high-dimensional feature representation which consists of three major modules: hypergraph generation, residual style connection and attention mechanism that learns the importance of heterogeneous relationships. How does these modules impact model performance? To answer this question, we conduct ablation studies to evaluate the performances of three model variants and Table 2 shows the experimental results. As can be seen, missing any of the three collaborators leads to performance degeneration.

4.3.3 Parameter Sensitivity and Runtime. The hyperparameter k, the number of nearest neighbors selected to be connected for generating hyper-graphs, influences the connectivity and the degree to which information is shared in the whole graph. Figure 4 shows trends in model performances as k changes. On the one hand, when k is too small, e.g., $k = 1$, the performance is unsatisfactory since the generated hyper-nodes bring about difficulties in accurate modeling and the edges between hyper-nodes do not afford to handle the issue. On the other hand, if we connect each pair of hyper-nodes and build the complete graph among them, the newly increased edges are in the majority which results in both bad performance and high cost for training as shown in Figure 4(c). When k is tens of orders of magnitude, we do not find significant differences in terms of both F1 score and AUC indeed.

4.3.4 Attention Coefficients. We further study the influence of different types of edges and the attention coefficients of invitation/IP/DID1/DID2 relationships are 0.1954/0.1048/0.4461/0.2537,



(a) F1 Performance (b) AUC Performance (c) Epoch Training Time

Figure 4: The influence of k in the procedure of hyper-graph generation. Note that $k = 1000$ indicates connecting each hyper-node with all the left hyper-nodes which results in complete graph among hyper-nodes.

respectively. The empirical studies show that the edges identified by the same physical device play the most significant role for fraud detection and the attention coefficient is 0.6998 in total. This is consistent with the fact that the fraudulent samples in the dataset are obtained mainly through the specific rules on the physical device.

4.3.5 Online Results. In practice, there are tens of thousands of newly registered users attracted by invitation in one business at iQIYI. We have deployed our method in the online production environment to detect fraudulent users. To make a more reliable prediction and minimize disruption to normal users, the threshold of identifying a user as a fraudster is set to 0.995. Online results show that it can cover 18.33% more suspicious accounts at the cost of a few customer complaints compared with the previous rule-based method. This improvement in the ability of risk management can bring into thousands and thousands of decrease of economic loss according to the business experience.

4.3.6 Further Discussion. As shown in Table 1 and discussed in Section 4.3.1, supervised methods sometimes outperform traditional GNNs. However, considering the practical issues in real businesses, GNNs are more reliable because the researchers usually need explaining the model predictions for either offline testing or consumers, especially when customer complaints come. Our model is a variant of GNNs and it not only shows safe predictions [11, 12] in a variety of performance metrics but also is more easily to give such explanations by mining the users in the connected subgraphs based on the relative domain knowledge.

5 CONCLUSION

This paper proposes HmGnn, a novel heterogeneous graph neural network, to detect fraudulent invitations at iQIYI platform. To the best of our knowledge, this is the first attempt to study fraud invitation detection via graph neural networks. HmGnn handles the homogeneity and heterogeneity of networks simultaneously. The proposal constructively introduces edges between homogenous mini-graphs based on the similarity of mini-graphs and a heterogeneous attention convolution network is then adopted to fuse the newly generated heterogeneous graphs. Extensive experiments conducted on real-world business data validate the excellent effectiveness and improvement on risk management of our method. In future, we are interested in extending our method to more practical applications, especially in the dynamic graph scenarios.

REFERENCES

[1] Qiang Cao, Xiaowei Yang, Jieqi Yu, and Christopher Palow. 2014. Uncovering Large Groups of Active Malicious Accounts in Online Social Networks. In *CCS*. 477–488.
[2] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation Learning for Attributed Multiplex Heterogeneous Network. In *SIGKDD*. 1358–1368.
[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *SIGKDD*. 785–794.
[4] Paul Erdós and Alfred Rényi. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hungary. Acad. Sci.* 5 (1960), 17–61.
[5] Qingyuan Gong, Jiayun Zhang, Yang Chen, Qi Li, Yu Xiao, Xin Wang, and Pan Hui. 2019. Detecting Malicious Accounts in Online Developer Communities Using Deep Learning. In *CIKM*. 1251–1260.
[6] Macro Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *IJCNN*, Vol. 2. 729–734.
[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. In *ECCV*. 630–645.
[8] Junxian Huang, Yinglian Xie, Fang Yu, Qifa Ke, Martín Abadi, Eliot Gillum, and Zhuoqing Morley Mao. 2013. SocialWatch: detection of online service abuse via large-scale social graphs. In *AsiaCCS*. 143–148.
[9] Wen-bing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. 2018. Adaptive Sampling Towards Fast Graph Representation Learning. In *NeurIPS*. 4563–4572.
[10] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[11] Yu-Feng Li, Lan-Zhe Guo, and Zhi-Hua Zhou. 2019. Towards Safe Weakly Supervised Learning. *TPAMI* (2019).
[12] Yu-Feng Li, Han-Wen Zha, and Zhi-Hua Zhou. 2017. Learning safe prediction for semi-supervised regression. In *AAAI*. 2217–2223.
[13] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. 2018. Heterogeneous Graph Neural Networks for Malicious Account Detection. In *CIKM*. 2077–2085.
[14] Meng Qu, Jian Tang, and Jiawei Han. 2018. Curriculum Learning for Heterogeneous Star Network Embedding via Deep Reinforcement Learning. In *WSDM*. 468–476.
[15] Gianluca Stringhini, Pierre Mourlanne, Grégoire Jacob, Manuel Egele, Christopher Kruegel, and Giovanni Vigna. 2015. Evilcohort: Detecting Communities of Malicious Accounts on Online Services. In *USENIX*. 563–578.
[16] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[17] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Peng Cui, P. Yu, and Yanfang Ye. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.
[18] Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrahy, Geoff Hulten, and Ivan Osipkov. 2008. Spamming botnets: signatures and characteristics. In *SIGCOMM*. 171–182.
[19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.