Enabling Graph Neural Networks for Semi-Supervised Risk Prediction in Online Credit Loan Services

HAO TANG, Tongji University, China
CHENG WANG*, Tongji University, China, the Key Laboratory of Embedded System and Service Computing Ministry of Education, China, and Shanghai Artificial Intelligence Laboratory, China
JIANGUO ZHENG, Tongji University, China
CHANGJUN JIANG, Tongji University, China, the Key Laboratory of Embedded System and Service Computing Ministry of Education, China, and Shanghai Artificial Intelligence Laboratory, China

Graph neural networks (GNNs) are playing exciting roles in the application scenarios where features are hidden in information associations. Fraud prediction of online credit loan services (OCLSs) is such a typical scenario. But it has another rather critical challenge, i.e., the scarcity of data labels. Fortunately, GNNs can also cope with this problem due to their good ability of semi-supervised learning by mining structure and feature information within graphs. Nevertheless, the gain of internal information is often too limited to help GNNs handle the extreme deficiency of labels with high performance beyond the basic requirement of fraud prediction in OCLSs. Therefore, adding labels from the experts, such as manually adding labels through rules, has become a logical practice. However, the existing rule engines for OCLSs have the confliction problem among continuously accumulated rules. To address this issue, we propose a Snorkel-based Semi-Supervised GNN (S3GNN). Under S3GNN, we specially design an upgraded version of the rule engines, called *Graph-Oriented Snorkel* (GOS), a graph-specific extension of Snorkel, a widely-used weakly supervised learning framework, to design rules by subject matter experts (SMEs) and resolve confliction. In particular, in the graph of anti-fraud scenario, each node pair may have multiple different types of edges, so we propose the *Multiple Edge-Types Based Attention* mechanism. In general, for the heterogeneous information and multiple relations in the graph, we first obtain the embedding of applicant nodes by aggregating the representation of attribute nodes, and then use the attention mechanism to aggregate neighbor nodes on multiple meta-paths to get ultimate applicant node embedding. We conduct experiments over the real-life data of a large financial platform. The results demonstrate that S3GNN can outperform the state-of-the-art methods, including the method of pilot platform.

CCS Concepts: • **Information systems** → **Information systems applications**; • **Theory of computation** → Design and analysis of algorithms.

Additional Key Words and Phrases: Fraud Prediction, Graph Neural Networks, Weak Supervision, Online Credit Loan Services

*Corresponding author.

Authors' addresses: Hao Tang, htangshh@gmail.com, Tongji University, China, 201804; Cheng Wang, cwang@tongji.edu.cn, Tongji University, China, 201804 and the Key Laboratory of Embedded System and Service Computing Ministry of Education, China, 201804 and Shanghai Artificial Intelligence Laboratory, China, 200030; Jianguo Zheng, zhengjianguo@tongji.edu.cn, Tongji University, China, 201804; Changjun Jiang, cjjiang@tongji.edu.cn, Tongji University, China, 201804 and the Key Laboratory of Embedded System and Service Computing Ministry of Education, China, 201804 and Shanghai Artificial Intelligence Laboratory, China, 200030.

2157-6904/2023/9-ART $15.00
https://doi.org/10.1145/3623401

1 INTRODUCTION

The online credit loan service (OCLS) is one of the most basic functional patterns of Internet Finance [56]. Beyond doubt, the lifeblood of OCLSs is the risk evaluation of bad debts. The bad debts, also called *bad loans*, refer to the loans that online lending platforms cannot recover from borrowers. They are usually of two types: (1) the loan whose borrower is unable to repay the due loan for certain reasons; (2) the loan whose beneficiary deliberately denies it. The latter is the *fraudulent loan*, i.e., the so-called *fraud*, where its beneficiary, i.e., the *fraudster*, usually uses the forged or misused information to defraud the loan [42]. Actually, OCLSs conduct the operation of "credit evaluation" to predict the non-fraud bad loans under the confirmation that the loans are not frauds. Therefore, fraud prediction is really necessary for predicting all kinds of bad loans at the pre-loan stage. It is exactly the focus of this work.

For loan fraudsters, the forged or misused application information should be used as many times as possible in order to defraud more money. This inevitably generates associations among application information of different loans. Then, fraud features often lurk in some of these associations. To a large extent, fraud detection is indeed to mine such associations. The challenges here come from two facts: (1) The implicit associations containing fraud information often hide deeply in extremely sparse and multidimensional association networks of application information; (2) most of on-going loans have no labels since OCLSs cannot determine whether they are frauds until a certain amount of non-payment. Traditional machine learning methods, e.g., SVM [50], deep learning [41], and tree-based methods [3], are not good at dealing with these two problems [54], though they have achieved qualified anti-fraud performance in other Internet financial scenarios.

To mine those deep associations, graph-based learning methods, e.g., the graph embeddings (GEs) [11] and graph neural networks (GNNs) [64], are almost certainly the preferred methods of choice. GEs, as a special case of *transductive* learning, are difficult to satisfy the low latency demand for OCLSs [9], since the embeddings need to be relearned whenever the graph is updated. Different from GEs, spatial-domain graph convolution networks, as one of *inductive* learning methods [12, 14, 34, 35, 60], do not need to be retrained every time new data comes, which are perfectly suited to fraud prediction by mining anomalous associations in real-time for OCLSs. As shown in Figure 1, it illustrates how to construct anti-fraud network based on the original submitted loan applications.

Before GNNs could carry out fraud prediction for OCLSs where data labels are grossly insufficient as mentioned above, they are confronted with another difficulty, i.e., semi-supervised learning problem. The good news is that GNNs can be naturally applied to the semi-supervised classification task on the graph [54]. Existing methods focus on improving GNNs' ability of semi-supervised learning by fully mining the internal information of graph data. Then, the performance improvement is logically limited by the quality of data, especially the credibility of labels. A semi-supervised learning method still requires a certain number of labels, otherwise the learned classifier will lack generalization capability due to the insufficient propagation of label information [47] which undermines the feasibility of GNNs to the fraud prediction problem in OCLSs.

To break through such a performance ceiling, a natural idea is to increase the amount of labeled data. Some common methods used to achieve this are label propagation [22] and SMOTE [7]. Label propagation algorithm struggles to handle unbalanced label distribution [61]. SMOTE exhibits a certain degree of randomness and suffers from distribution marginalization issues [43]. As a straightforward solution, the preliminary review by the subject matter experts (SMEs) can provide more effective and accurate information from which the rough labels of unlabeled data can be obtained. Specifically, a potentially effective SME-based method is *active learning* [44]. It selects the most informative part of unlabeled data, and gives them to the SMEs to label, then adds these labeled data to the training set and repeats the above steps. By active learning, a pretty good performance model can be built. The core of active learning is to make up for the lack of label quantity by selecting the most valuable data. As a matter of fact, active learning is relatively applicable to the scenarios where the experts only need the

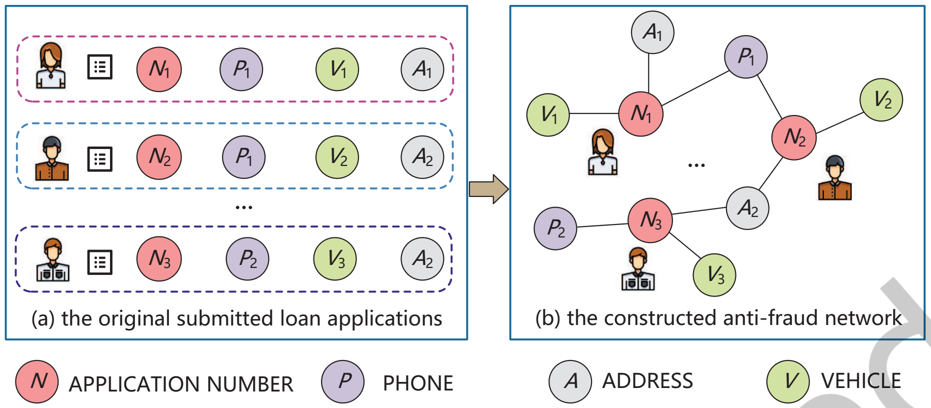

(a) the original submitted loan applications
(b) the constructed anti-fraud network
N APPLICATION NUMBER
P PHONE
A ADDRESS
V VEHICLE

Fig. 1. An example of constructing anti-fraud network from the original submitted loan applications, where a loan application consists of the unique identifier and three different attributes. All N_i, $i = 1, 2, 3$, represent applicant numbers, which are identifiers of loan applications. All P_i, A_i, and V_i, $i = 1, 2, 3$, represent different attribute nodes, respectively. The submitted original applications on the financial client platform are shown in Figure 1(a). It is difficult for the financial client platforms to evaluate the risk of loan applications adequately due to losing associations between these loan applications. In Figure 1(b), the anti-fraud network is constructed from three original submitted loan applications. The risk of loan applications can be analyzed more accurately by mining associations between the applications.

ability with some concrete knowledge, e.g., the ability to classify given pictures into cats and dogs. Nevertheless, it has a significant disadvantage especially for fraud prediction in OCLSs. It heavily depends on the anti-fraud expertise of SMEs due to the requirement of nonintuitive judgment and abstract knowledge for fraud prediction.

This work turns to anti-fraud *rules* to increase data labels, since SMEs have been designing a lot of rules based on the experience and formed *rule engines* [15]. Actually, compared with labeling training data, in real-life lending platforms, rules are more often adopted to build rule-based fraud prediction engines with black lists, where the main obstacle is the more and more serious confliction among continuously accumulated rules. So this work is devoted to exploring *how to use rules to increase labeled training data for boosting GNNs in fraud prediction of OCLSs.*

We first propose an effective solution based on a weak supervision framework, i.e., Snorkel [40], then call it *Snorkel-based Semi-Supervised Graph Neural Network* (S3GNN). Particularly, we specially devise a dedicated version of rule engines, called *Graph-Oriented Snorkel* (GOS), a graph-specific extension of Snorkel. Like rule-based engines, Snorkel also utilizes SMEs' knowledge to design rules— or call it the process of writing labeling functions. But a more important advance is that Snorkel adopts a generative model to resolve contradictions or dependencies among rules, while our GOS further considers the graphical constraints. With the data labeled by GOS, we train the subsequent heterogeneous attention-based GNN. It's important to point out that in the anti-fraud graph, there may be multiple types of edges between each node pair. Therefore, it is necessary for each node to learn a unique representation from different types of edges. To address this challenge, we propose the *Multiple Edge-Types Based Attention* to fusion the information of different edge types. It is worth mentioning that, to our best knowledge, we are the first to apply the Snorkel framework to graph data and make improvements to adapt to graphs.

To evaluate our S3GNN, we conduct the experiments on the real-life dataset. It is demonstrated that S3GNN has superiority in fraud prediction of OCLSs.

The contributions of our work are summarized as follows:

(1) We introduce external knowledge into GNNs and achieve the cooperation of internal and external information of graphs, which provides a new perspective for solving semi-supervised problems in graph representation learning.

(2) We customize Snorkel specifically for graph data and learning to devise a graph-oriented Snorkel (GOS).

(3) We evaluate our S3GNN on the real-life dataset. The experiments demonstrate that S3GNN has superiority in fraud prediction of OCLSs.

The rest of the paper is structured as follows: We summarize related works in Section II; Section III presents preliminary and the detail of our S3GNN model; Section IV shows experiment results; Finally, Section V concludes the paper and envisage future work.

2 RELATED WORK

2.1 Anti-Fraud in Online Services

Cyber fraud has become a serious social and economic problem [26]. Machine learning, as a promising data-driven paradigm, has been widely used in anti-fraud fields [51] for various network services by supervised learning approaches like logistic regression, SVM or neural networks [3, 41, 50]. Babaev *et al.* [1] adopted a deep learning method, RNNs, on fine-grained transnational data to compute credit scores for the loan applicants, and produced significant financial gains for the banks. Suarez-Tangil *et al.* [46] combined a range of structured, unstructured, and deep-learned features to build a detection system which stopped scammers as they create fraudulent profiles or before they engaged with potential victims.

Aforementioned methods regard each entity and extract features from different aspects. However, in financial scenarios, entities may have rich interactions with each other. Then a few works start to utilize the graph for fraud detection. Wang *et al.* [53] extracted fine-grained co-occurrence relationships by using a knowledge graph and realized data enhancement for diversified behavior models in online payment fraud detection. Li *et al.* [29] proposed a method for modeling the transactions by using a multipartite graph, and detected the complete flow of money from source to destination in money laundering criminal detection. Liang *et al.* [31] developed an automated solution for fraud detection based on graph learning algorithms to separate fraudsters from regular customers and uncover groups of organized fraudsters in insurance fraud detection. Cheng *et al.* [10] investigated the problem of predicting repayment delinquency in the networked-guarantee loans, and attempted on predicting bank guarantee loan defaults by adding objectively temporal network structure using an end-to-end learning framework.

Recently, several works start to use graph neural networks in fraud detection. GEM [36] adaptively learned embeddings from heterogeneous graphs based on two prime weaknesses of attackers for malicious account detection. SemiGNN [54] proposed a hierarchical attention mechanism to ensure the results for similar nodes similar by proposing the graph-based loss. GRC [60] uses self-attention mechanism to characterize multiple types of relationships, and uses conditional random fields to ensure users with the same role have similar representations. GAGA [57] offers a flexible way to deal with the problem of low homophily in graphs for fraud detection tasks. DAGNN [28] uses two aggregation strategies to augment the disparity of heterogenous neighbors and augment the similarity to homogeneous neighbors, respectively. AO-GNN [20] overcomes the challenges of imbalanced labels and noisy graph structure for fraud detection by focusing on AUC maximization. Graph anomaly detection is a key technique for graph fraud detection, and recent advances have been made in graph anomaly detection. BWGNN [49] employs spectral and spatial localized band-pass filters to better cope with the ‘right-shift’ phenomenon in the detection of graph anomalies. Dgraph [21] is constructed with a variety of novel and promising properties for graph anomaly detection. GraphBERT [58] learns comprehensive representations of tweets and users to detect malicious behaviors.

2.2 Graph Neural Networks

Graph neural networks (GNNs) can effectively extract the structure information and node information of the graph to learn better node representation. Based on the spectrum theory, Burna *et al.* [5] applied the learnable convolution operation to the graph for the first time. GCN [24] simplified the definition of spectrum-based graph convolution. GraphSAGE [14] applied the aggregation operator to gather the neighbor information to achieve the inductive learning. GAT [52] applied the attention mechanism to the graph neural network for the first time.

All these work above are for homogeneous graphs, but most real-world graphs are heterogeneous information networks with multi-typed nodes and edges. HAN [55] used two-level attention mechanism in heterogeneous graphs and used meta-paths to transform a heterogeneous graph into homogeneous graphs. HetGNN [63] used sampling strategy based on randomwalk with restart to convert heterogeneous graphs into homogeneous graphs. Both HetSANN [17] and GTN [62] explored to directly encode node information in heterogeneous graphs without using manually designed meta-paths. xFraud [39] trains a self-attention heterogeneous graph neural network through malicious transactions, and outputs human-understandable explanations through the explainer, which has taken a big step in the interpretability of the graph model.

However, although most work mentions that GNNs can be applied to semi-supervised problems, only Sun *et al.* [47] mentioned that too few labels will limit the performance ceiling of GNN.

2.3 Weak Supervision

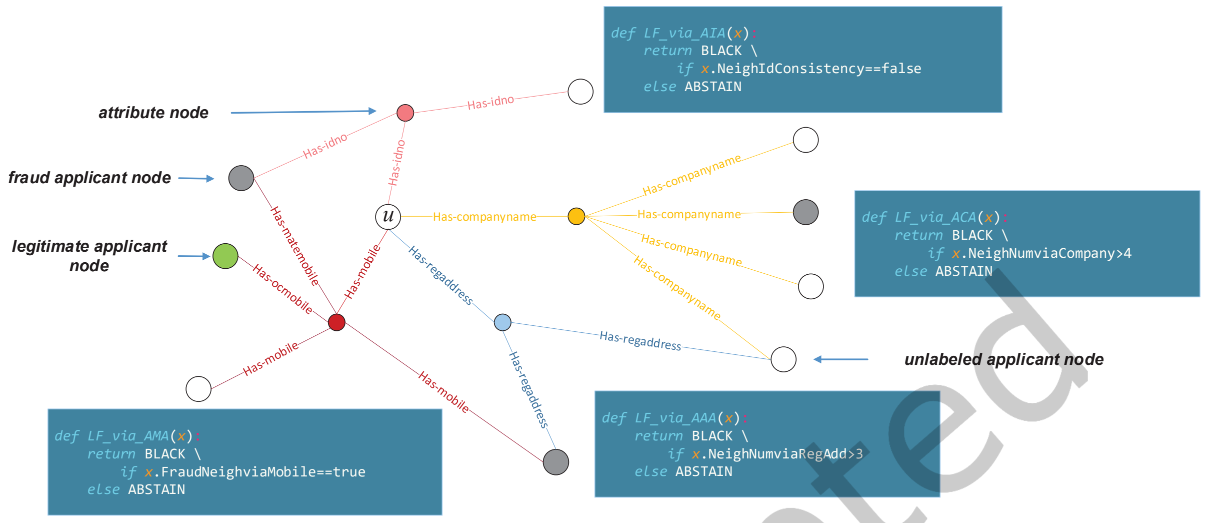
Weak supervision can be roughly divided into three categories [65]: incomplete supervision, inexact supervision and inaccurate supervision. We mainly focus on incomplete supervision. Semi-supervised learning [6] took some assumptions to heuristically label the unlabeled data. Active learning [44] aimed to label the most informative data for the model with experts interventions. Recently, there have been many works focused on weak supervision in fraud risk detection. SPFD [48] proposed constrained seed k-means for cash pre-loan fraud detection. MFEFD [19] was trained in a semi-supervised manner which combined the supervised and unsupervised training for detecting electricity fraud. As a semi-supervised method, GCN [24] can adapt to insufficient labels, although this will affect its performance limit.

Snorkel [40] is a weak supervision framework that can quickly produce labeled training data. It can quickly produce, manage, and model training data by using labeling functions. Wu *et al.* [59] provided a new programming model that enables users to convert domain expertise, based on multiple modalities of information to enhance the input of Snorkel. Fries *et al.* [13] presented a method for training weakly supervised medical entity classifiers using off-the-shelf ontologies as a source of reusable and easily automated labeling heuristics. This method mitigates the limitations of Snorkel about deploying labeling heuristics in the medical domain.

However, Snorkel is not specifically for graphs, and the existing methods seldom consider the constraints of edges in graphs and neglect the associations among the graph structure.

3 THE PROPOSED METHOD

To overcome the significant weakness of labels, we propose a novel method for risk prediction in online lending services, called *Snorkel-based Semi-Supervised Graph Neural Network* (S3GNN). The overview of whole method is shown in Figure 3. Our method is divided into four steps. For an applicant associated graph full of heterogeneous information, the first step is to design meta-paths based labeling functions from expert knowledge, and apply these functions to label some unlabeled nodes. In the second step, we train the generative model with graph constraints and get rough labels of unlabeled nodes. Third, we train the customized heterogeneous graph neural



```
def LF_via_AIA(x):
    return BLACK \
        if x.NeighIdConsistency==false
    else ABSTAIN
```

```
def LF_via_ACA(x):
    return BLACK \
        if x.NeighNumviaCompany>4
    else ABSTAIN
```

```
def LF_via_AMA(x):
    return BLACK \
        if x.FraudNeighviaMobile==true
    else ABSTAIN
```

```
def LF_via_AAA(x):
    return BLACK \
        if x.NeighNumviaRegAdd>3
    else ABSTAIN
```

Fig. 2. The design of labeling functions. This is a representative partial subgraph extracted from our real-world data set. We hide the specific attribute values and retain the main topological relationships. The central applicant node u is connected to other applicant nodes through multiple attribute nodes. We observe that these application nodes are associated with node u via different meta-paths, and each meta-path contains different rule information. In this partial subgraph, there are mainly four meta-paths (represented by edges in different colors): Applicant-CompanyName-Applicant (ACA), Applicant-Address-Applicant(AAA), Applicant-Mobile-Applicant(AMA), Applicant-Idno-Applicant(AIA). We design these simple labeling functions on different meta-paths based on some effective expert knowledge.

network to learn the embedding of applicant nodes on the entire graph. Finally, we use the focal loss to train the heterogeneous graph model and aggregate embeddings on all meta-paths. We also describe the detailed operation process of our scheme in Algorithm 1.

3.1 Preliminary

3.1.1 Heterogeneous Graph. A heterogeneous graph contains either multiple types of nodes or multiple types of edges, which is defined as $G = (V, E)$, V denotes nodes and E denotes edges in the graph.

Example. In the heterogeneous graph of IMDb network, as shown in Figure 4(a), there is only one edge type between *Actor* and *Movie*. In our heterogeneous graph for applicant information associated network, the type of edge between nodes is not only determined by the types of node pairs. As shown in Figure 4(b), with the meta-path *Applicant-Address-Applicant*, although two applicant nodes are connected to the same attribute node *Address*, the relationship categories of the two edges are *has-HomeAddress-of* and *has-Compaddress-of*. The nodes V are mainly divided into two types, applicant nodes U and attribute nodes T, among which attribute nodes can be divided into more types. Each applicant node is connected to the attribute nodes through different types of edges. So it is necessary to add multiple edge-types to consider the cross-correlation between applicant nodes.

3.1.2 Meta-path. A meta-path Υ [55] is defined as a path in the form of

$$A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \dots \xrightarrow{R_l} A_{l+1},$$

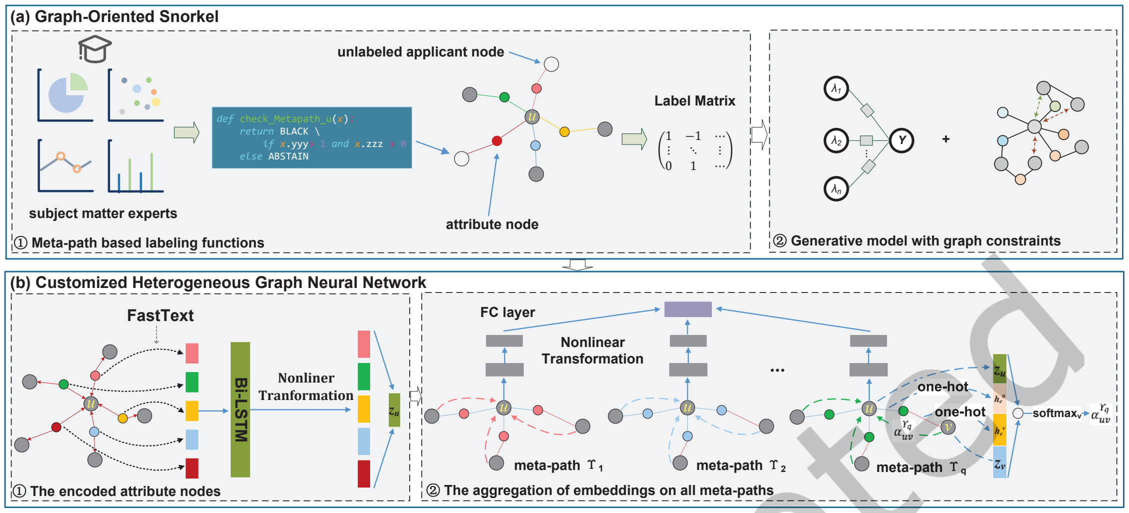

(a) Graph-Oriented Snorkel
unlabeled applicant node
Label Matrix
subject matter experts
attribute node
① Meta-path based labeling functions
② Generative model with graph constraints
(b) Customized Heterogeneous Graph Neural Network
FastText
Bi-LSTM
Nonliner Tranformation
FC layer
Nonlinear Transformation
one-hot
softmax
meta-path Τ1
meta-path Τ2
meta-path Τq
① The encoded attribute nodes
② The aggregation of embeddings on all meta-paths

Fig. 3. The framework of the proposed model S3GNN. a-① shows how to design labeling functions on different meta-paths and generate a label matrix. a-② tells how to train the generative model with graph constraints and get rough labels of unlabeled nodes. b-① describes the process of encoding the attribute nodes and initializing the embedding of the applicant node (node u in this case) by the aggregation of its neighbor attribute nodes and deep features. b-② shows how to aggregate the embeddings of neighbor nodes through attention mechanism on different meta-paths through fully connected layer.

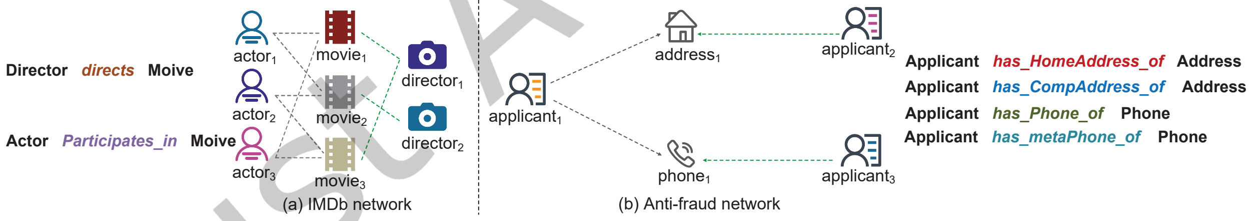

Director directs Moive
Actor Participates_in Moive
actor1
actor2
actor3
movie1
movie2
movie3
director1
director2
(a) IMDb network
applicant1
address1
applicant2
phone1
applicant3
(b) Anti-fraud network
Applicant has_HomeAddress_of Address
Applicant has_CompAddress_of Address
Applicant has_Phone_of Phone
Applicant has_metaPhone_of Phone

Fig. 4. The difference between the anti-fraud network and the IMDb network. (a) The types of edges in the heterogeneous graph of IMDb network are determined by the node pairs; (b) There are more types of edges in the anti-fraud heterogeneous graph, which are not only determined by the node pairs. For the same node pair, the edges can also be a more fine-grained type.

abbreviated as $A_1A_2 \ldots A_{l+1}$, which describes a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between objects A_1 and A_{l+1}, where $\circ$ denotes the composition operator on relations.

3.1.3 Labeling Function. Rather than hand-labeling training data, users of Snorkel write labeling functions, which allow them to express various weak supervision sources such as external knowledge bases, heuristics, and more.

3.1.4 Generative Model. Since the labeling functions we design are all empirical rules, they must contain some contradictory or interdependent relationships. The original Snorkel models the true class label for a data point as a latent variable in a probabilistic model. For example, we model each labeling function as a noisy voter which is independent, i.e., makes errors that are uncorrelated with the other labeling functions. We can also model statistical dependencies between the labeling functions to improve predictive performance. For example, if two labeling functions express similar heuristics, we can include this dependency in the model and avoid a double counting problem. Then we adopt structure learning method of Snorkel for selecting a set C of labeling function pairs (j, k) to model as correlated [2].

This defines a generative model of the votes of the labeling functions as noisy signals about the true label. The original Snorkel encodes a factor graph to integrate these noisy signals by using three factor types to represent the labeling propensity, accuracy, and pairwise correlations of labeling functions [40]:

$$\phi_j^{Prop}(\Lambda_i, y_i) = \mathit{1}\{\Lambda_{i,j} \neq \emptyset\}, \tag{1}$$

$$\phi_j^{Acc}(\Lambda_i, y_i) = \mathit{1}\{\Lambda_{i,j} = y_i\}, \tag{2}$$

$$\phi_{j,k}^{Corr}(\Lambda_i, y_i) = \mathit{1}\{\Lambda_{i,j} = \Lambda_{i,k}\}, \quad (j, k) \in C, \tag{3}$$

where $\Lambda_i \in \mathbb{R}^n$ is the labeling vector of an unlabeled graph node x_i. Note that the true label y_i of x_i is a latent variable that can't be observed. For the unlabeled graph node x_i, Snorkel defines the vector $\phi(\Lambda_i, y_i)$ as the concatenation of these factors over all labeling functions $j = 1, \ldots, n$ and potential correlations C, and the corresponding vector is denoted by $\theta \in \mathbb{R}^{2n+|C|}$. For general binary classification, the generative model $P(\Lambda_i, y_i)$ is defined as a joint distribution:

$$P(\Lambda_i, y_i) = \frac{\exp\Big(\theta^T \phi(\Lambda_i, y_i)\Big)}{\sum_{y_i \in \{0,1\}} \exp\Big(\theta^T \phi(\Lambda_i, y_i)\Big)}. \tag{4}$$

3.2 Graph-Oriented Snorkel

The designed GOS is a weak supervision framework that can efficiently absorb expert knowledge. The original design of Snorkel [40] is not specifically for graphs. Considering the constraints of edges in graphs, we modify its generative model accordingly. GOS provides effectual ways to construct labeling functions. From the evaluation results of each node by the labeling functions, a label matrix is generated. By a generative model with graphical constraints, we can get the rough labels for unlabeled nodes.

3.2.1 Writing Meta-path Based Labeling Functions. We generate training labels by writing labeling functions. The labeling function takes an unlabeled node $x \in X$ as input and returns a label, formally, $\lambda : X \rightarrow Y \bigcup \{\emptyset\}$, where $Y \in \{0, 1\}$ as binary setting and $\emptyset$ denotes that the labeling functions abstains.

Our graph of loan applicants contains a variety of textual information. It is a really professional problem to use this information to write labeling functions. We design the labeling functions for each meta-path in the heterogeneous graph.

In this method, we treat the heterogeneous graph as multiple homogeneous graphs related by meta-paths. Since each meta-path contains corresponding business information and expert rules, we mine these expert rules in each meta-path and design the corresponding labeling functions.

For example, on the meta-path of *applyno-address-applyno*, the two applicant nodes are connected to an address node and the edges are the *has-companyaddress* and *has-homeaddress*. Based on expert experience, the fraudsters usually use misused application information or even use illegal means to obtain other people's information and then impersonate others' identities to apply for loan applications (as shown in Figure 5), and both applications are at risk of fraud because of the same address exhibits two different functional attributes.

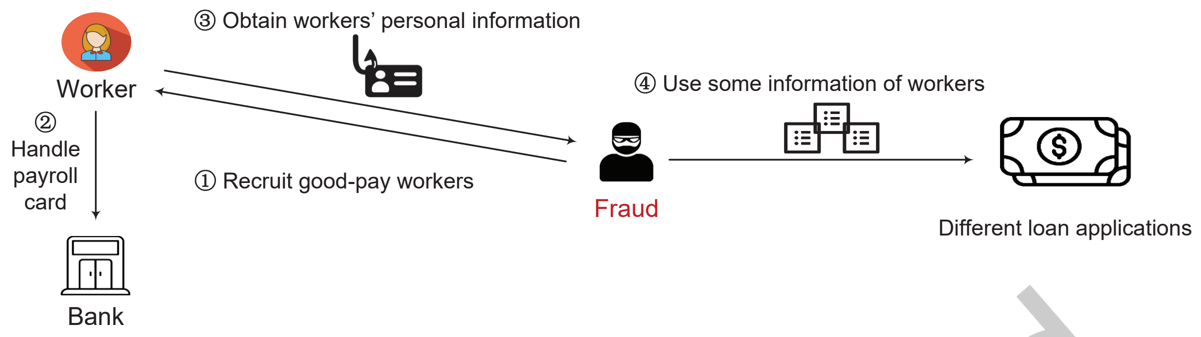

③ Obtain workers' personal information
Worker
② Handle payroll card
Bank
① Recruit good-pay workers
④ Use some information of workers
Fraud
Different loan applications

Fig. 5. A representative example of how fraudsters collect information from victims. The fraud group pretends to be a legitimate agency under the guise of recruiting good-pay click farming workers (Figure 5, ①). The fraud group gives each worker a mobile phone card and asks them to apply for a payroll card in the bank (Figure 5, ②). Under the pretext of registration, the fraudsters use bank cards and mobile phone numbers to obtain the worker's ID card, home address, and other information (Figure 5, ③). Finally, the fraudsters use some information of workers to apply for a number of loan applications on the online loan platform by binding the payroll card (Figure 5, ④).

For another example, as a labeling function $LF\_via\_ACA(x)$ shown in Figure 2, multiple loan applicants all claim to have a company, but they have the same company name. Obviously, this is impossible even if judged by common sense, so we determine that these applicants constituted gang fraud. Note that each labeling function can only cover a part of data, which is the reason why we need to use expert knowledge to design right number of labeling functions.

We can modify the labeling function according to the specific requirements. Here is an example of a labeling function in Python3:

```python
def check_companyaddress(x):
    return BLACK \
        if x.addedge_type == 'home'\
            and x.addedge_neightype == 'company'\
            and x.oneorder_fraud > 1
    else ABSTAIN
```

The code is explained as follows. If an edge type of an applicant node is *has-homeaddress* and the other end of the edge is a *company* attribute node, and the number of fraud nodes in the first-order neighbor of the applicant node is greater than 1, then we determine that the application node is a fraud node, otherwise we will abstain.

Given m unlabeled graph nodes and n labeling functions, we can generate a label matrix $\Lambda^{m\times n}$ based on voting results of all labeling functions for all unlabeled nodes, where $\Lambda_{i,j} = \lambda_j(x_i)$ represents the result of labeling function j for the unlabeled node x_i. If labeling function i for the unlabeled node j abstained, the value of $\Lambda_{i,j}$ is -1.

3.2.2 Generative Model with Graph Constraints. As a matter of fact, the generative model of Snorkel is not designed for graph-oriented applications. Although it considers the potential correlations between labeling functions in the generative model, it neglects the correlations between data points. Such correlations are precisely what makes graph data different from other types of data.

According to the main features of frauds in OCLSs, we consider the importance of the structure between nodes for the energy function. Therefore, we define the weight matrix W:

$$w_{ij} = \begin{cases} \frac{\exp\left(-\frac{\|\Lambda_i - \Lambda_j\|^2}{\delta^2}\right)}{\sqrt{hop_{ij}}} & \text{if } x_j \in N_k(x_i), \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where δ is a hyperparameter and hop_{ij} is the number of hops between x_i and x_j, and $N_k(x_i)$ denotes the set of k-order neighbors of x_i. The constraint on all graph nodes makes the energy function pay more attention to those nodes that are close to the target node and have similar features. Inspired by [66], we introduce an energy function $E(\Lambda)$ to measure the label similarity between unlabeled nodes on the entire graph:

$$E(\Lambda) = \sum_{i,j} \sum_{y_i, y_j \in \{0,1\}} w_{ij} \left(\frac{P(\Lambda_i, y_i)}{P(\Lambda_i)} - \frac{P(\Lambda_j, y_j)}{P(\Lambda_j)} \right)^2 [y_i = y_j], \tag{6}$$

$$P(\Lambda_i) = \sum_{y_i \in \{0,1\}} P(\Lambda_i, y_i). \tag{7}$$

Here $P(\Lambda_i)$ is the marginal distribution, and $[y_i = y_j]$ is Iverson's convention form [25]. Then we learns the parameter $\theta \in \mathbb{R}^{2n+|C|}$ by minimizing the negative log marginal likelihood to avoid using the true labels y:

$$\widehat{\theta} = \arg\min_{\theta} -\sum_{i=1}^{m} \log P(\Lambda_i) + \epsilon E(\Lambda). \tag{8}$$

Let ϵ control the contributions of graph constraints. Here $-\sum_{i=1}^{m} \log P(\Lambda_i)$ denote the negative loglikelihood to find the parameters that are most likely (i.e., with the greatest probability) to cause the samples distribution, and $\epsilon E(\Lambda)$ reduces the complexity of the model by adding graph constraints that encourage parameters that the closer nodes are, the more likely they share the same feature distribution. We convert the outputs of the generative model $\widetilde{y_i} = P_{\widehat{\theta}}(y_i|\Lambda_i)$ into rough labels and these rough labels can be used to a discriminative model:

$$y_i = \begin{cases} 1 & \text{if } \widetilde{y_i} > 0.5, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

3.3 Customized Heterogeneous Graph Neural Network

In this paper we represent the applicant information associated graph as a heterogeneous graph. A heterogeneous graph is defined as $G = (V, E)$, V denotes nodes and E denotes edges in the graph. The nodes V are mainly divided into two types, applicant nodes U and attribute nodes T, among which attribute nodes can be divided into more types. Each applicant node is connected to the attribute nodes through different types of edges.

3.3.1 Feature Engineering. Considering the anti-fraud field of online lending, traditional machine learning methods in the past rely heavily on manual features constructed by experts. Therefore, here we adopt a combination of a small number of manual features and features generated by the explicit automated feature engineering method, i.e., Deep Feature Synthesis (DFS) [23], to reduce the time consumed by manpower and enhance the utility of features.

It should be pointed out that the so-called manual features are mainly some of the graph features that we extracted from the graph. After storing our heterogeneous graph in the Neo4j database, we use Cypher to quickly

Algorithm 1: The process of S3GNN

```
Input: The heterogeneous graph $G = (V, E)$,
    The meta-path set $\{\Upsilon_1, \Upsilon_2, \cdots, \Upsilon_q\}$,
    The number of multi-head attention $K$,
    Meta-path based labeling functions $\{LF_1, LF_2, \cdots, LF_n\}$.
Output: The anti-fraud model $M_g$.
Generate labels with Graph-Oriented Snorkel:
// the system generates labels for these unlabeled nodes
if the label matrix has not been generated then
    The system generates labels for these unlabeled nodes
    for each unlabeled node $i = 1, 2, \ldots, m$ do
        for each labeling function $j = 1, 2, \ldots, n$ do
            get the annotation value $\lambda_{ij}$;
        get the $LF$ annotation vector $\lambda_i$;
    get the label matrix $\lambda$.
else
    Train the generative model with constraint
    $\widehat{\theta} = \arg\min_{\theta} -\sum_{i=1}^{m} \log P(\Lambda_i) + \epsilon E(\Lambda)$ and obtain rough labels.
Train customized heterogeneous graph neural network:
// encode attribute nodes and applicant nodes
for $t \in T$ do
    Utilize FastText and DeepWalk to generate $h_t$.
for $u \in U$ do
    Compute $h_u$ and $h_{DFS}$ to concatenate $z_u = h_u \oplus h_{DFS}$.
// train the anti-fraud model
for $\Upsilon \in \{\Upsilon_1, \Upsilon_2, \cdots, \Upsilon_q\}$ do
    for $i \in V$ do
        Find the meta-path based neighbors $N_u^{\Upsilon}$;
        for $j \in N_u^{\Upsilon}$ do
            Compute the weight coefficient $\alpha_{uv}^{\Upsilon}$
        Compute $z_u^{\Upsilon} = \bigoplus_{k=1}^{K} \sigma\left(\sum_{v \in N_u^{\Upsilon}} \alpha_{uv}^{\Upsilon} \cdot z_v\right)$;
    Aggregate edge-types based embedding $z_u = \frac{1}{q} \sum_{l=1}^{q} \sigma(W \cdot z_u^{\Upsilon_l} + b)$.
Update the model parameters
return the anti-fraud model $M_g$
```

query the statistics of nodes or edges to generate graph features. Table 1 shows the main graph features we extracted.

The next step is to enable automated feature engineering and use deep synthesis algorithms to automatically construct other risk features as input to the risk model. The Deep Feature Synthesis (DFS) [23] is an algorithm for automatic feature construction. Its input has two main parts. One of the inputs is a collection of related

Table 1. The extracted features in loan transactions.

Features	**Description**
Degree	The number of neighbors of the node.
ApplOne	The number of first-order neighbors of the node.
ApplTwo	The number of second-order neighbors of the node.
FraudOne	The number of fraud neighbors in the node's first-order neighbor.
FraudTwo	The number of fraud neighbors in the node's second-order neighbor

entities, and the other input is a basic element function, that is, a series of mathematical functions. There are many calculation functions required to construct a feature, and these calculation functions can be divided into two types of calculation functions, i.e., basic conversion function and basic aggregate function. We denote them as $F = (f_{a_1}, f_{a_2}, \dots, f_{b_1}, f_{b_2}, \dots)$, where a and b are different types of basic element function; f_{a_1} is one certain calculation function in a basic element function. The basic aggregate function takes a column of features of related instances in one or more entity tables as input, and returns a value at output, which is equivalent to the aggregate function of a relational database, such as an Average function, a Median function, and so on. The basic conversion function takes a column or certain columns of features of all instances in an entity table as input, and returns a new column of features at output, which is equivalent to a common operator in a relational database, such as an Addition function, an Equal function and so on.

Algorithms based on deep feature synthesis will generate a large number of features, so we filter the features. We use Pearson correlation coefficient to measure the linear correlation between features:

$$pcc = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}},$$

the Pearson correlation coefficient usually refers to the quotient of the covariance and standard deviation between two variables. The Pearson correlation coefficient is widely used in measuring the linear correlation of two vectors. Its value varies from -1 to + 1.If the value is +1, it means that the two vectors are positively correlated, a value of -1 means a negative correlation, and a value of 0 means that there is no linear correlation. After that, we apply feature clustering and aggregate features in different clusters. For each feature f_i in the aggregated feature set, we calculate the correlation pcc between it with the target feature f_t. If $pcc_{i,t} < \epsilon_f$, this feature will be removed. According to the principle that the larger the standard deviation, the closer the filter threshold is to the average value, the filter threshold ϵ_f is calculated by:

$$\epsilon_f = E(pcc_{i,t})^{-\frac{1}{D(pcc_{i,t})}}, \tag{10}$$

where $E(pcc_{i,t})$ and $D(pcc_{i,t})$ are the mean expectation and variance of correlation between candidate feature f_i and target feature f_t, respectively. This step achieves the purpose of removing the features that are unrelated or weakly correlated with the target variable. We denote the feature vector of the applicant node $u \in U$ obtained in this step as h_{DFS}.

3.3.2 Encoding Attribute Nodes and Applicant Nodes. Although the automated feature engineering has obtained some useful features, in order to make full use of the attribute information contained in the nodes, we encode these attribute nodes. The attribute nodes contain text information or numerical features. For example, the *address* node contains the family, household or company address and the *company* node contains the company name and a brief introduction. The *idno* node has the identity card number. For the attribute node $t \in T$ which has text information, we utilize FastText [4] to pre-train the text information and we denotes the embedding of t as h_t. For the attribute node t which has numerical features, we use the DeepWalk [38] to generate the walk sequence of the same type of attribute node and then generate the embedding of the attribute node as $h_t \in \mathbb{R}^{d_h \times 1}$.

After we have the embedding of the attribute nodes, there are several choices in how to represent the applicant nodes according to the attribute embedding. For the correlations between attribute nodes, directly using simple concatenation or averaging for attribute embedding might lose these correlations. Inspired by HetGNN [63], we utilize bi-directional LSTM (Bi-LSTM) [16] to capture the deep interaction information between attribute nodes. Given an applicant node $u \in U$ and its neighbor attribute nodes T_u, its initial embedding $z_u \in \mathbb{R}^{d \times 1}$ is calculated as follows:

$$h_u = \frac{1}{|T_u|} \cdot \sum\nolimits_{t \in T_u} tanh(\overrightarrow{LSTM}(h_t) \oplus \overleftarrow{LSTM}(h_t)), \tag{11}$$

$$z_u = h_u \oplus h_{DFS}, \tag{12}$$

where $\oplus$ denotes the concatenation. After the deep interaction of Bi-LSTM and nonlinear transformation, we can better capture the correlations between the attribute nodes and achieve deeper integration.

3.3.3 Multiple Edge-Types based Attention. The previous graph neural networks usually study graphs with a single edge type between nodes [54, 55] and the influence of multiplex edge types is rarely considered. For example, in the IMDb network, as shown in Figure 4(a), there is only one edge type between *Actor* and *Movie*. In our applicant information associated network, the type of edge between nodes is not only determined by the types of node pairs. As shown in Figure 4(b), with the meta-path *Applicant-Address-Applicant*, although two applicant nodes are connected to the same attribute node *Address*, the relationship categories of the two edges are *has-HomeAddress-of* and *has-Compaddress-of*. So it is necessary to add Multiple Edge-Types based Attention (MEA) to consider the cross-correlation between applicant nodes.

Given an applicant node $u \in U$ and a meta-path Υ, where $\Upsilon \in \{\Upsilon_1, \Upsilon_2, \cdots, \Upsilon_q\}$, we denote N_u^{Υ} as the meta-path based neighbors of the node u (include itself). Since the number of edge types is fixed, we use one-hot to encode the edges between the applicant nodes and the attribute nodes. For example, given a pair of applicant nodes u and v, there are two edges e_u and e_v connecting from node u and node v to the same attribute node, respectively. We denote h_e^u, h_e^v as the one-hot embedding of e_u and e_v, respectively. Then the attention score α_{uv}^{Υ} between nodes u and v is formulated as follows:

$$e_{uv}^{\Upsilon} = att(z_u, z_v, h_e^u, h_e^v), \qquad \alpha_{uv}^{\Upsilon} = \frac{exp(e_{uv}^{\Upsilon})}{\sum_{k \in N_u^{\Upsilon}} exp(e_{uk}^{\Upsilon})}, \tag{13}$$

where the attention mechanism att is calculated as:

$$att(z_u, z_v, h_e^u, h_e^v) = LeakyReLU(a_{\Upsilon}^T [z_u \oplus z_v \oplus h_e^u \oplus h_e^v]). \tag{14}$$

Here a_{Υ}^T is a learnable vector. Now we can get the edge-types based embedding of node u which is denoted as z_u^{Υ}:

$$z_u^{\Upsilon} = \sigma\left(\sum\nolimits_{v \in N_u^{\Upsilon}} \alpha_{uv}^{\Upsilon} \cdot z_v\right), \tag{15}$$

To capture more information from different representation subspaces, we use multi-head attention that includes K independent edge-types based attentions. Then we concatenate the K embeddings as edge-types based embedding:

$$z_u^{\Upsilon} = \overset{K}{\underset{k=1}{\oplus}} \sigma\left(\sum\nolimits_{v \in N_u^{\Upsilon}} \alpha_{uv}^{\Upsilon} \cdot z_v\right). \tag{16}$$

Table 2. The selected nodes in application transactions.

Nodes	**Attribute**	**Description**
Application	Name, Time	The identifier of a transaction. We extract the applicant's name and the time of the transaction as attributes.
ADDR	Province,City, District	The detailed address. We extract three levels of administrative areas as attribute description.
CONAME	None	The name of the company where the online loan applicant works.
INDO	Name	The identity card number. Use holder's name as an attribute.
DL	None	The driving license plate number of the applicant.
VIN	None	The vehicle identification number of the loan applicant.
ENGINE	None	The engine number of the loan applicant.
MP	Name	The mobile phone number. Use holder's name as an attribute.
TEL	Name	The telephone number. Use holder's name as an attribute.

We get the corresponding set of embeddings $\{z_u^{\Upsilon_1}, z_u^{\Upsilon_2}, \cdots, z_u^{\Upsilon_q}\}$ for each meta-path in the set of meta-paths $\{\Upsilon_1, \Upsilon_2, \cdots, \Upsilon_q\}$ of the constructed heterogeneous graph.

3.3.4 Aggregating Edge-Types Based Embedding. For a given set of meta-paths $\{\Upsilon_1, \Upsilon_2, \cdots, \Upsilon_q\}$, we have a corresponding set of embeddings $\{z_u^{\Upsilon_1}, z_u^{\Upsilon_2}, \cdots, z_u^{\Upsilon_q}\}$. The final applicant node embedding is represented by this set of edge-types based embeddings. Then the final node embedding is calculated as:

$$z_u = \frac{1}{q} \sum_{l=1}^{q} \sigma(W \cdot z_u^{\Upsilon_l} + b), \tag{17}$$

where W is the weight matrix, b is the bias vector, q is the semanticlevel attention vector. Note that for the meaningful comparison, all above parameters are shared for all meta-paths and semantic-specific embedding.

3.4 Loss Function

In the traditional graph node classification tasks, the most commonly used loss function is cross entropy. But in the field of online lending and anti-fraud, our data distribution is extremely uneven. Fraud data accounts for only a small part of all data. Therefore, the original cross-entropy function is no longer suitable. For all labeled nodes, we use *Focal Loss* [33] to solve the unbalanced problem:

$$a_t = \begin{cases} a & if\ y = 1 \\ 1 - a & otherwise, \end{cases} \tag{18}$$

$$p_t = \begin{cases} p & if\ y = 1 \\ 1 - p & otherewise, \end{cases} \tag{19}$$

$$p = \sigma(\Theta \cdot z_u), \tag{20}$$

$$Loss = -a_t(1 - p_t)^{\gamma} \log(p_t), \tag{21}$$

Table 3. The extracted edges in loan transactions.

Edges	**Description**
R_ADDR	The loan applicant's address.
R_CO	The mobile number of the applicant's colleagues.
R_CO_ADDR	The address of the applicant's company.
R_CO_NAME	The loan applicant's company name.
R_CO_TEL	The telephone number of the applicant's company.
R_CPADDR	The loan applicant's residential address.
R_CRMP	The mobile phone number of the applicant's general relatives.
R_FCMP	The mobile phone number of the applicant's friends.
R_IDNO	The loan applicant's license plate number.
R_DL	The loan applicant's driving license plate number.
R_VIN	The vehicle identification number of loan applicant's car.
R_ENGINE	The engine number of loan applicant's car.
R_LRMP	The mobile phone number of the applicant's immediate family members.
R_MATEIDNO	The identity card number of the applicant's spouse.
R_MP	The loan applicant's mobile phone number.
R_OCMP	The mobile phone number of the applicant's other contacts.
R_TEL	The loan applicant's telephone number.
R_REGADDR	The loan applicant's native place.

where Θ is a learnable parameter, $a \in [0, 1]$ is a weighting factor, $p \in [0, 1]$ is the model's estimated probability for the class with label $y = 1$ and γ is a focusing parameter.

4 EXPERIMENTS

4.1 Datasets and Evaluation Metrics

4.1.1 Datasets. Our data set comes from a large-scale Internet financial lending platform. We select all the data from January 1, 2017 to July 31, 2017. After excluding isolated samples, there are a total of 86,019 application records. In order to avoid the problem of time crossing, we select the records from January 1 to June 30, 2017 as the training set, and the data from July 1 to July 31, 2017 as the test set. The heterogeneous graph built with these data contains 9 types of nodes (1 type of applicant node and 8 types of attribute nodes, as shown in Table 2) and 19 types of edges (as shown in Table 3). The division of datasets is shown in Table 4. Restricted by the privacy protection policy, the characteristics of the attribute nodes are all string types, and the ID number and phone

Table 4. The division of dataset and the number of labels.

Original Labels	**Ground-truth Labels**		**No Labels**		**Total**
Labels after GOS	**Legitimate**	**Fraudulent**	**No Labels**	**Generated Labels**	
Training	7751	1249	59789	1779	70568
Valid	1787	206	0	0	1993
Test	1692	219	11547	0	13458

fields are all desensitized. In order to facilitate the storage of large-scale network structures, we choose Neo4j graph database to store heterogeneous graphs.

4.1.2 Metrics. There are many evaluation metrics for general binary classification problems, such as AUC, F-measure and so on. However, the objective is not only to predict more fraud records, but also to reduce prediction errors as much as possible in an unbalanced data scenario such as anti-fraud. Therefore, we require a high recall rate and a low disturbance rate. To evaluate the performance of our method, we use three widely-used metrics in anti-fraud for OCLSs including Recall [45] (True Positive Rate), Disturbance [27] (False Positive Rate), and KS value [37] (Kolmogorov-Smirnov) that is calculated by KS= |max(TPR-FPR)|. The higher Recall reflects the better ability of the system to detect fraud. The lower the Disturbance is, the less impact it will have on normal applications. The definition of prediction is not black or white (discrete type) in the field of anti-fraud, so it is more reasonable to measure it with probability distribution (continuous type). KS value balances the models' ability to catch more frauds (high Recall) and reduce the effects on normal loan applications (low Disturbance), which is the de facto standard in the field of loan anti-fraud [18, 54].

4.2 Baseline Methods

We use eight baseline methods as follows:

- **XGBoost** [8]: It's a model that has been widely used and has achieved good results in industry recently. We use the embedding of the applicant node after aggregating neighbor attribute nodes as the input of the model.
- **ASNE+MLP** [32]: It is a model of attributed graph embedding which can simultaneously learn structural information and attribute information. We use MLP as the following classifier.
- **GCN** [24]:It is a model that propagates the information of neighbor nodes to node itself through convolution operation. We test all the meta-paths for GCN and report the best performance among different meta-paths.
- **GAT** [52]: It is an improved method which uses the attention mechanism. We test all the meta-paths for GAT and report the best performance among different meta-paths.
- **HetGNN** [63]: It uses a strategy of random walk with restart for node sampling instead of setting meta-paths.
- **SemiGNN** [54]: It is designed for financial fraud detection by applying a hierarchical attention mechanism to correlate different neighbors and different views. It considers the similarity of representations between nearby nodes in the design of the loss function.
- **LIFE** [30]: It conducts embedding learning of both nodes and edges in the heterogeneous network by a directed heterogeneous tripartite graph with attributed nodes and edges, and builds a fully-connected layer on top of the concatenation of all the embeddings.

Table 5. Qantitative results (%), **S+Baseline** is the combined version of baseline method and **GOS**.

Metrics	XGBoost	ASNE+MLP	GCN	GAT	HetGNN	SemiGNN	LIFE	S2GNN
KS ↑	34.5	37.7	39.1	41.6	44.6	45.2	44.1	**45.4**
Recall ↑	46.3	49.8	52.1	54.8	55.2	**64.5**	55.3	57.1
Disturbance ↓	11.8	12.1	13.0	13.2	10.6	**19.3**	11.2	11.7
Metrics	**S+XGBoost**	**S+ASNE+MLP**	**S+GCN**	**S+GAT**	**S+HetGNN**	**S+SemiGNN**	**S+LIFE**	**S3GNN**
KS ↑	38.6	39.2	42.5	44.9	47.4	47.8	47.1	**48.2**
Recall ↑	49.9	51.5	55.3	58.4	58.9	**64.7**	59.3	63.0
Disturbance ↓	11.3	12.3	12.8	13.5	11.5	**16.9**	12.2	14.8

- **S2GNN**: It is a reduced version of S3GNN that removes Graph-Oriented Snorkel.

4.3 Implementation Details

In the experiment, we repeat the test 5 times for each model and report the average performance. For our S3GNN, the embedding dimension of FastText [4] and edge-types based attention are set to 256, 128, respectively. The number of attention head K is 4. We optimize S3GNN with Adam and the learning rate, dropout rate are set to 0.01, 0.35, respectively. The number of meta-paths is 8. The length of Bi-LSTM is variable because the number of neighbor attribute nodes of the applicant node is not fixed [63]. The weighting factor a and focusing parameter γ are set to 0.2, 2. Then we use early stopping with a patience of 50. To select the most meaningful meta-path and fuse the information for GCN/GAT, we select each meta-path in the set of meta-paths of the multiplex heterogeneous graph to transfer it to different homogenous graphs and presented the best performance among these meta-paths after applying GCN/GAT. Our experiments are conducted on Windows Server 2012 with Intel Xeon E5-2640 v4 and 128 GB of RAM, and the code has been released[1].

4.4 Performance Comparison

The results are shown in table 5 and KS curves are shown in Figure 6. From them, we can get the following analysis:

- The performance of our proposed method S3GNN is better than other methods, which proves the superiority of our model. In the second stage of the online lending flow, the user data we collected is quite preliminary, and the KS value of 0.482[2] is already a pretty good performance especially in the absence of labels and only six months of data.
- The KS values of the S+ baselines have all been improved which proves GOS is effective to strengthen the label of the dataset. The KS value of S2GNN is 2.8% lower than that of S3GNN, which also proves that the introduction of GOS has brought better performance improvements to the model.

[1]Code link: https://github.com/MrZealand/S3GNN

[2]For a real-world online lending platform, according to the de facto standards, the ex-ante fraud prediction can be adopted as a qualified one if it has a KS value that is not much less than 0.5. But this standard is achieved when the platform has a considerable number of fraud samples. The lending platform who provided us with the data has accumulated seven years of data labels to achieve the goal of exceeding the KS value of 0.5.

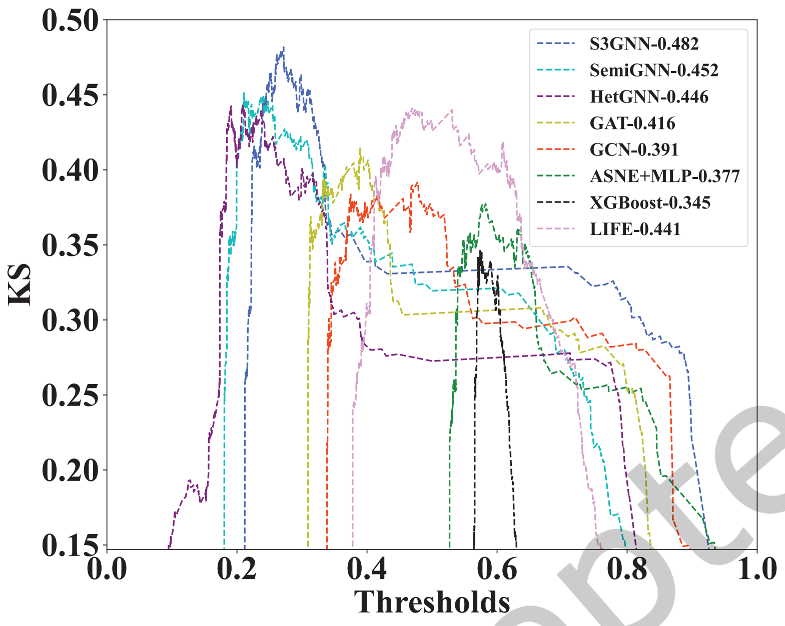
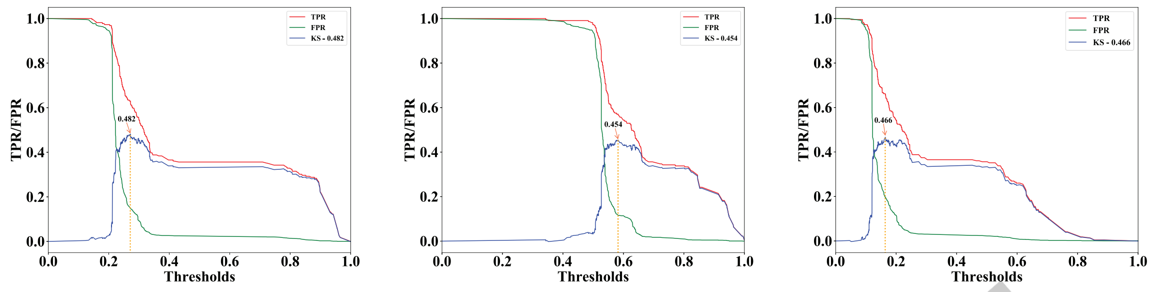


Fig. 6. The KS curves of S3GNN and baseline methods.

- SemiGNN achieves the highest Recall that demonstrates the effectiveness of its loss function design. Its loss function assumes that the nodes having fraud neighbors are also suspicious. If the embeddings of nodes that have fraud nodes are distinct from the embeddings of nearby fraud nodes, the output of loss function will increase, and we impose punishment on this situation which does not meet the expectation of loss design of SemiGNN. As a result, it ensures the embeddings of normal nodes that adjacent to fraud nodes are similar to the embeddings of fraud nodes and causes the misclassification of some normal nodes. So SemiGNN brings a higher disturbance compared with our method. The better performance of S2GNN than HetGNN proves the necessity of considering unbalanced data distribution.
- The performance of LIFE is not very good for OCLSs because it conducts embedding learning of both nodes and edges in the heterogeneous network directly and ignores the importance of meta-paths by using graph structure. In our method, after obtaining the embedding of applicant nodes by aggregating the representation of attribute nodes, we further use the attention mechanism to aggregate neighbor nodes on multiple meta-paths to get ultimate applicant node embedding.
- The performance of GCN and GAT are not good, which proves that some related information is lost after the heterogeneous graph converted into homogeneous graph. The comprehensive utilization of node information on multiple meta-paths is still an effective method.
- ASNE, as a traditional graph representation learning method, has lower performance than graph neural network methods. It proves the importance of obtaining information from neighbor nodes.
- XGBoost has the worst performance, probably because of lacking enough labels information and poor processing of attribute embeddings.



(a) S3GNN (b) S2GNN (c) S3GNN-noGC

Fig. 7. The TPR/FPR curves of S3GNN, S2GNN and S3GNN-noGC.

Table 6. Qantitative results (%), **Baseline-DFS** is the version of baseline method with **DFS** removed and doesn't include the input features of **DFS**, and **Baseline-MEA** is the version of baseline method without Multiple Edge-types based Attention.

Metrics	**XGBoost-DFS**	**ASNE+MLP-DFS**	**GCN-DFS**	**GAT-DFS**	**HetGNN-DFS**	**SemiGNN-DFS**	**LIFE-DFS**	**S2GNN-DFS**	**S2GNN-DFS-MEA**	**S2GNN-MEA**
KS ↑	12.5	14.4	20.3	21.2	22.3	26.4	22.1	28.3	23.1	**44.5**
Recall ↑	39.4	38.7	41.1	42.7	44.9	48.3	45.2	46.6	44.2	**56.5**
Disturbance ↓	**26.9**	24.3	20.8	21.5	22.6	21.9	23.1	18.3	21.1	12.0
Metrics	**S+XGBoost-DFS**	**S+ASNE+MLP-DFS**	**S+GCN-DFS**	**S+GAT-DFS**	**S+HetGNN-DFS**	**S+SemiGNN-DFS**	**S+LIFE-DFS**	**S3GNN-DFS**	**S3GNN-DFS-MEA**	**S3GNN-MEA**
KS ↑	13.2	14.7	22.6	22.1	23.0	27.9	23.3	30.4	26.8	**46.2**
Recall ↑	39.5	39.3	41.7	43.5	47.1	49.4	48.5	49.2	46.3	**61.9**
Disturbance ↓	**26.3**	24.6	19.1	21.4	24.1	21.5	25.2	18.8	19.5	15.7

4.5 Ablation Study

To verify the effects of the main components of our method, we conduct a comprehensive evaluation from three aspects, i.e., graph constraints (GC), deep feature synthesis (DFS) and multiple edge-types based attention (MEA). The results is shown in Table 6.

- **GC**: Since graph constraints (GC) are added to the generative model based on the initial Snorkel, we explore the impact of graph constraints on model performance, and the results are shown in Figure 7. S3GNN-noGC is the version which includes meta-path based labeling functions and excludes generative model with graph constraints, and the KS value is 46.6%. S2GNN is the version that has neither meta-path based labeling functions nor generative models with graph constraints, and the KS value is 45.4%. The KS value of S3GNN-noGC is 1.2% higher than that of S2GNN, which proves that the effectiveness of meta-path based labeling functions, and it can mitigate deficiency of labels to some extent. The KS value of S3GNN-noGC is 1.6% lower than that of S3GNN, which proves that graph constraints are also important for the characteristics of labels distribution in graph.

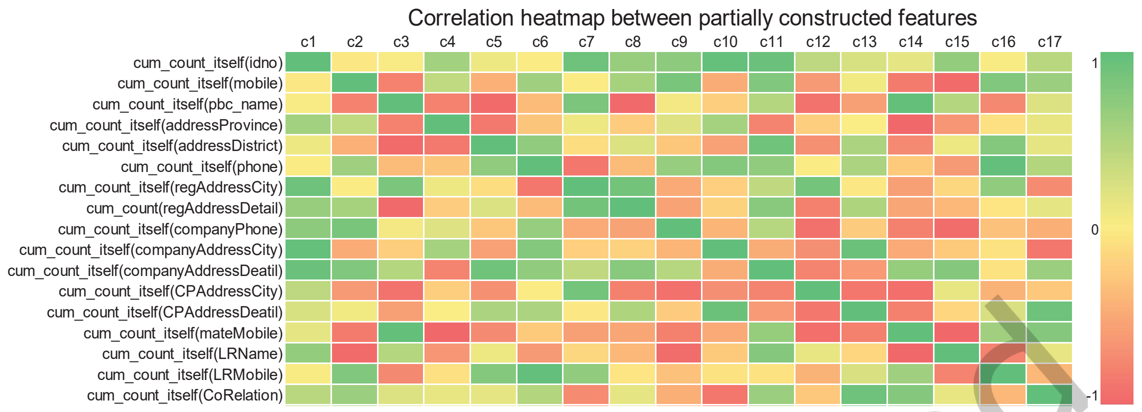


Fig. 8. Correaltion heatmap between partially constucted features.

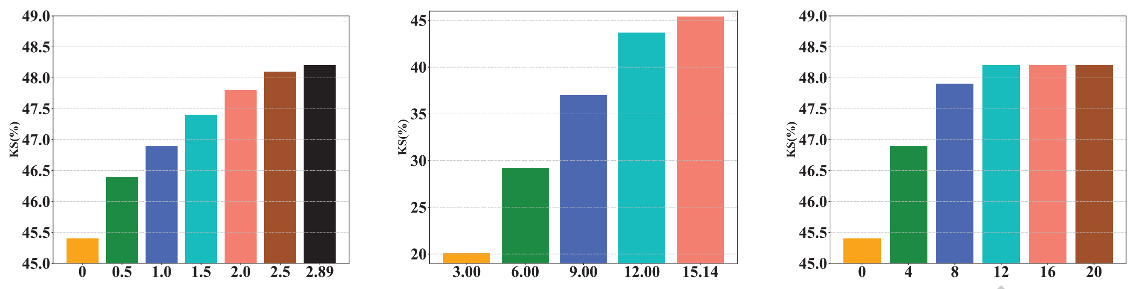
- **DFS**: In order to explore the influence of manual features and the deep features generated based on this on the experiment, we design a comparative experiment. In the feature generation stage, we constructed 2793 features, and 222 features are left after feature selection by density clustering algorithm. Figure 8 shows the heat map of the correlation between 17 deep features constructed in the DFS algorithm. The greener the color, the higher the correlation. Then we use density clustering to cluster the features corresponding to the green color into a cluster, which will be retained for subsequent tasks. From the results in Table 5, it can be seen that after removing the features generated by the DFS algorithm, the performance of all models has a cliff-like decline. It proves that in the anti-fraud task, we cannot rely solely on the attribute information of the graph nodes. The graph relationship features and the deep features generated based on this are also of importance.
- **MEA**: To explore the impact of multiple edge-types based attention on model performance, we trained a model without using multiple edge-types based attention (MEA). We observed a degradation in performance after removing the MEA, indicating that the multiple edge-types between node pairs in the heterogeneous anti-fraud network are crucial for preserving important graph structural information. Therefore, ignoring these edge-types can negatively impact model performance.

4.6 Parameter Sensitivity

Among the hyperparameters of S3GNN, the number of rough labels generated by GOS is undoubtedly the most important. Due to the limited expert information we obtained, the coverage rate of labeling functions is also relatively low. Since the proportion of fraud and normal data is unbalanced, GOS outputs all rough fraud labels, and this part accounts for 2.89% of the total data. We set up different proportions of rough fraud labels, and show the experimental results in Figure 9(a). It shows that as the number of rough fraud nodes increases, the KS value also increases significantly .

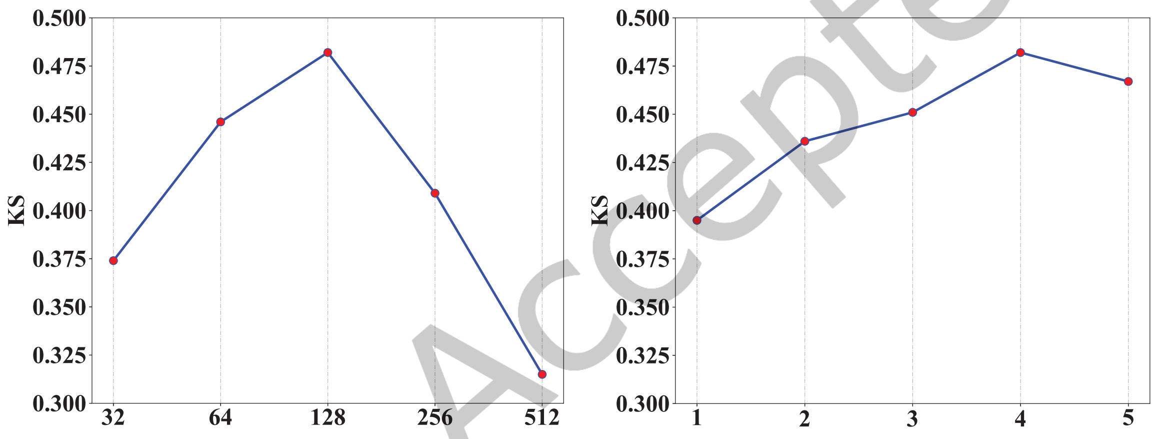
Figure 9(b) shows the performance comparison of S2GNN under different true label proportions. It can be seen that as the number of true labels increases, the performance increase of S2GNN is decreasing, which shows that the number of labels is not linearly related to model performance. However, S3GNN increases the number of rough fraud labels by less than 3%, which alleviates the uneven distribution of fraud and normal data. The performance of S3GNN is greatly improved compared with S2GNN.

Figure 9(c) illustrates the impact of the number of label functions on performance. It can be observed that as the number of label functions increases, the performance increase quickly encounters a bottleneck. This is



(a) The proportions of rough labels (b) The proportions of true labels (c) The number of LFs

Fig. 9. (a) Performance of S3GNN on different proportions of rough labels from Snorkel. (b) Performance of S2GNN on different proportions of ture labels. (c) Performance of S3GNN on different numbers of labeling functions.



(a) The dimension of edge-types based attention (b) The number of attention head K

Fig. 10. Parameter sensitivity includes the dimension of edge-type based embedding and the number of attention head.

because the data covered by the label functions we later design is highly overlapped with the previous label functions and cannot bring performance improvement. If we can have more expert knowledge from the lending platform, the performance will be more better.

We also conducted experiments on other hyperparameters including the dimension of edge-type based embedding and the number of attention head. As shown in Figure 10(a), the best performance is achieved when the dimension is 128. But when the dimension further increases, the performance begins to drop sharply. It leads to overfitting when the dimension is too large. As shown in Figure 10(b), the model achieves the best performance when the number of head is 4. Excessive number of attention head will also weaken the performance of the model. Attention heads of an appropriate number are enough to capture different semantic information and make model stable.

5 CONCLUSION

This work proposes a Snorkel-based semi-supervised graph neural network model for fraud prediction in online credit loan services (OCLSs). In the case that existing GNNs have reached the limit of their performance by mining the internal information of the graph, we utilize a Graph-Oriented Snorkel to absorb external expert knowledge to improve the performance ceiling in the scenarios of fewer labels. Then we design a heterogeneous graph neural network based on the attention mechanism. In particular, when calculating the embeddings of applicant nodes, we consider the impact of different types of multiple edge-types. Our method achieves better experimental results than the baseline method, and the results demonstrate the effectiveness of meta-path based labeling functions and generative model with graph constraints. Furthermore, our experiments verify the importance of manual features and deep features. For future work, we will further focus on the timing evolution of heterogeneous graph in OCLSs.

ACKNOWLEDGEMENT

The authors sincerely thank the anonymous referees for their helpful comments and suggestions. The research of authors is partially supported by the National Natural Science Foundation of China (NSFC) under Grant 61972287, the Special Security Foundation of Shanghai Municipal Commission of Economy and Information Technology under Grant No. 214102, the Fundamental Research Funds for the Central Universities under Grant 22120210524, the open fund of Key Laboratory of Industrial Internet of Things and Networked Control, Ministry of Education, under Grant 2021FF08, and the Program of Shanghai Academic Research Leader under Grant 22XD1423700.

REFERENCES

[1] Dmitrii Babaev, Maxim Savchenko, Alexander Tuzhilin, and Dmitrii Umerenkov. 2019. ET-RNN: Applying Deep Learning to Credit Loan Applications. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2183–2190.

[2] Stephen H Bach, Bryan He, Alexander Ratner, and Christopher Ré. 2017. Learning the structure of generative models without labeled data. In *International Conference on Machine Learning*. PMLR, 273–282.

[3] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J Christopher Westland. 2011. Data mining for credit card fraud: A comparative study. *Decision Support Systems* 50, 3 (2011), 602–613.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral Networks and Locally Connected Networks on Graphs. *arXiv:1312.6203* (2013).

[6] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks* 20, 3 (2009), 542–542.

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[8] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *ACM KDD 2016*. 785–794.

[9] Dawei Cheng, Zhibin Niu, Yi Tu, and Liqing Zhang. 2018. Prediction defaults for networked-guarantee loans. In *IEEE ICPR 2018*. 361–366.

[10] Dawei Cheng, Yiyi Zhang, Fangzhou Yang, Yi Tu, Zhibin Niu, and Liqing Zhang. 2019. A dynamic default prediction framework for networked-guarantee loans. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2547–2555.

[11] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2018. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering* 31, 5 (2018), 833–852.

[12] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. 2020. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 315–324.

[13] Jason A Fries, Ethan Steinberg, Saelig Khattar, Scott L Fleming, Jose Posada, Alison Callahan, and Nigam H Shah. 2021. Ontology-driven weak supervision for clinical entity classification in electronic health records. *Nature communications* 12, 1 (2021), 1–11.

[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[15] Frederick Hayes-Roth. 1985. Rule-based systems. *Commun. ACM* 28, 9 (1985), 921–932.

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[17] Huiting Hong, Hantao Guo, Yucheng Lin, Xiaoqing Yang, Zang Li, and Jieping Ye. 2020. An Attention-Based Graph Neural Network for Heterogeneous Structural Learning.. In *AAAI 2020*. 4132–4139.

[18] Binbin Hu, Zhiqiang Zhang, Jun Zhou, Jingli Fang, Quanhui Jia, Yanming Fang, Quan Yu, and Yuan Qi. 2020. Loan default analysis with multiplex graph learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2525–2532.

[19] Tianyu Hu, Qinglai Guo, Xinwei Shen, Hongbin Sun, Rongli Wu, and Haoning Xi. 2019. Utilizing unlabeled data to detect electricity fraud in AMI: A semisupervised deep learning approach. *IEEE Transactions on Neural Networks and Learning Systems* 30, 11 (2019), 3287–3299.

[20] Mengda Huang, Yang Liu, Xiang Ao, Kuan Li, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. Auc-oriented graph neural network for fraud detection. In *Proceedings of the ACM Web Conference 2022*. 1311–1321.

[21] Xuanwen Huang, Yang Yang, Yang Wang, Chunping Wang, Zhisheng Zhang, Jiarong Xu, Lei Chen, and Michalis Vazirgiannis. 2022. Dgraph: A large-scale financial dataset for graph anomaly detection. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[22] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. 2019. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5070–5079.

[23] James Max Kanter and Kalyan Veeramachaneni. 2015. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, 1–10.

[24] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv:1609.02907* (2016).

[25] Donald E Knuth. 1992. Two notes on notation. *The American Mathematical Monthly* 99, 5 (1992), 403–422.

[26] Sangho Lee and Jong Kim. 2013. Warningbird: A near real-time detection system for suspicious urls in twitter stream. *IEEE transactions on dependable and secure computing* 10, 3 (2013), 183–195.

[27] Yok Yong Lee, Mohd Hisham Dato Haji Yahya, Muzafar Shah Habibullah, and Zariyawati Mohd Ashhari. 2019. Non-performing loans in European Union: country governance dimensions. *Journal of Financial Economic Policy* (2019).

[28] Qiutong Li, Yanshen He, Cong Xu, Feng Wu, Jianliang Gao, and Zhao Li. 2022. Dual-Augment Graph Neural Network for Fraud Detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4188–4192.

[29] Xiangfeng Li, Shenghua Liu, Zifeng Li, Xiaotian Han, Chuan Shi, Bryan Hooi, He Huang, and Xueqi Cheng. 2020. FlowScope: Spotting Money Laundering Based on Graphs.. In *AAAI*. 4731–4738.

[30] Zhao Li, Haishuai Wang, Peng Zhang, Pengrui Hui, Jiaming Huang, Jian Liao, Ji Zhang, and Jiajun Bu. 2021. Live-streaming fraud detection: a heterogeneous graph neural network approach. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3670–3678.

[31] Chen Liang, Ziqi Liu, Bin Liu, Jun Zhou, Xiaolong Li, Shuang Yang, and Yuan Qi. 2019. Uncovering Insurance Fraud Conspiracy with Network Learning. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1181–1184.

[32] Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2018. Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2257–2270.

[33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *ICCV 2017*. 2980–2988.

[34] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2021. Pick and Choose: A GNN-based Imbalanced Learning Approach for Fraud Detection. In *Proceedings of the Web Conference 2021*. 3168–3177.

[35] Yang Liu, Xiang Ao, Qiwei Zhong, Jinghua Feng, Jiayu Tang, and Qing He. 2020. Alike and Unlike: Resolving Class Imbalance Problem in Financial Credit Risk Assessment. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2125–2128.

[36] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. 2018. Heterogeneous graph neural networks for malicious account detection. In *ACM CIKM*. 2077–2085.

[37] Xiangfeng Meng, Yunhai Tong, Xinhai Liu, Yiren Chen, and Shaohua Tan. 2017. Netrating: Credit risk evaluation for loan guarantee chain in china. In *Pacific-Asia Workshop on Intelligence and Security Informatics*. Springer, 99–108.

[38] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[39] Susie Xi Rao, Shuai Zhang, Zhichao Han, Zitao Zhang, Wei Min, Zhiyao Chen, Yinan Shan, Yang Zhao, and Ce Zhang. 2020. xFraud: Explainable Fraud Transaction Detection on Heterogeneous Graphs. *arXiv preprint arXiv:2011.12193* (2020).

[40] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *VLDB 2017*, Vol. 11. 269.

[41] Pediredla Ravisankar, Vadlamani Ravi, G Raghava Rao, and Indranil Bose. 2011. Detection of financial statement fraud and feature selection using data mining techniques. *Decision support systems* 50, 2 (2011), 491–500.

[42] Ke Ren and Avinash Malik. 2019. Investment Recommendation System for Low-Liquidity Online Peer to Peer Lending (P2PL) Marketplaces. In *ACM WSDM 2019*. 510–518.

[43] José A Sáez, Julián Luengo, Jerzy Stefanowski, and Francisco Herrera. 2015. SMOTE–IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Information Sciences* 291 (2015), 184–203.

[44] Burr Settles. 2009. *Active learning literature survey*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences.

[45] Mohammad Ahmad Sheikh, Amit Kumar Goel, and Tapas Kumar. 2020. An approach for prediction of loan approval using machine learning algorithm. In *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*. IEEE, 490–494.

[46] Guillermo Suarez-Tangil, Matthew Edwards, Claudia Peersman, Gianluca Stringhini, Awais Rashid, and Monica Whitty. 2019. Automatically dismantling online dating fraud. *IEEE Transactions on Information Forensics and Security* 15 (2019), 1128–1137.

[47] Ke Sun, Zhouchen Lin, and Zhanxing Zhu. 2020. Multi-Stage Self-Supervised Learning for Graph Convolutional Networks on Graphs with Few Labeled Nodes.. In *AAAI 2020*. 5892–5899.

[48] Wanlin Sun, Ming Chen, Jie-xia Ye, Yuhang Zhang, Cheng-zhong Xu, Yangqing Zhang, Yaonan Wang, Wen Wu, Peng Zhang, and Feipeng Qu. 2019. Semi-Supervised Anti-Fraud Models for Cash Pre-Loan in Internet Consumer Finance. In *ICPS 2019*. 635–640.

[49] Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. 2022. Rethinking graph neural networks for anomaly detection. In *International Conference on Machine Learning*. PMLR, 21076–21089.

[50] Jun Tang and Jian Yin. 2005. Developing an intelligent data discriminating system of anti-money laundering based on SVM. In *ICMLC 2005*, Vol. 6. 3453–3457.

[51] Liang Tong, Bo Li, Chen Hajaj, Chaowei Xiao, Ning Zhang, and Yevgeniy Vorobeychik. 2019. Improving robustness of {ML} classifiers against realizable evasion attacks using conserved features. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 285–302.

[52] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv:1710.10903* (2017).

[53] Cheng Wang and Hangyu Zhu. 2020, DOI: 10.1109/TDSC.2020.2991872. Representing Fine-Grained Co-Occurrences for Behavior-Based Fraud Detection in Online Payment Services. *IEEE Transactions on Dependable and Secure Computing* (2020, DOI: 10.1109/TDSC.2020.2991872).

[54] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. 2019. A Semi-supervised Graph Attentive Network for Financial Fraud Detection. In *IEEE ICDM 2019*. 598–607.

[55] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW 2019*. 2022–2032.

[56] Yang Wang, Wenchun Wang, Jiaojiao Wang, et al. 2017. Credit risk management framework for rural commercial banks in China. *Journal of Financial Risk Management* 6, 01 (2017), 48.

[57] Yuchen Wang, Jinghui Zhang, Zhengjie Huang, Weibin Li, Shikun Feng, Ziheng Ma, Yu Sun, Dianhai Yu, Fang Dong, Jiahui Jin, et al. 2023. Label Information Enhanced Fraud Detection against Low Homophily in Graphs. *arXiv preprint arXiv:2302.10407* (2023).

[58] Jiele Wu, Chunhui Zhang, Zheyuan Liu, Erchi Zhang, Steven Wilson, and Chuxu Zhang. 2022. GraphBERT: Bridging Graph and Text for Malicious Behavior Detection on Social Media. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 548–557.

[59] Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. 2018. Fonduer: Knowledge base construction from richly formatted data. In *Proceedings of the 2018 international conference on management of data, SIGMOD*. 1301–1316.

[60] Bingbing Xu, Huawei Shen, Bingjie Sun, Rong An, Qi Cao, and Xueqi Cheng. 2021. Towards Consumer Loan Fraud Detection: Graph Neural Networks with Role-Constrained Conditional Random Field. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4537–4545.

[61] Yuto Yamaguchi and Kohei Hayashi. 2017. When Does Label Propagation Fail? A View from a Network Generative Model.. In *IJCAI*. 3224–3230.

[62] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. In *Advances in Neural Information Processing Systems*. 11983–11993.

[63] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *ACM KDD 2019*. 793–803.

[64] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv:1812.08434* (2018).

[65] Zhi-Hua Zhou. 2018. A brief introduction to weakly supervised learning. *National Science Review* 5, 1 (2018), 44–53.

[66] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003*. 912–919.