frontiers | Frontiers in Physics

TYPE Original Research
PUBLISHED 09 May 2023
DOI 10.3389/fphy.2023.1196294

OPEN ACCESS

EDITED BY
Fei Yu,
Changsha University of Science and Technology, China

REVIEWED BY
Dongfan Chao,
Hainan University, China
Xiong Li,
East China Jiaotong University, China

*CORRESPONDENCE
Lei Chen,
chenlei@hnust.edu.cn

RECEIVED 29 March 2023
ACCEPTED 24 April 2023
PUBLISHED 09 May 2023

CITATION
Lei Y, Chen L, Li Y, Xiao R and Liu Z (2023), Robust and fast representation learning for heterogeneous information networks. *Front. Phys.* 11:1196294. doi: 10.3389/fphy.2023.1196294

Robust and fast representation learning for heterogeneous information networks

Yong Lei[1,2], Lei Chen[3]*, Yuan Li[3], Ruifeng Xiao[4] and Zhaohua Liu[3]

[1]School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China, [2]Hunan Key Laboratory for Service Computing and Novel Software Technology, Xiangtan, China, [3]School of Information and Electrical Engineering, Hunan University of Science and Technology, Xiangtan, China, [4]School of Computer Science and Railway Engineering, Hunan Technical College of Railway High-speed, Hengyang, China

Network representation learning is an important tool that can be used to optimize the speed and performance of downstream analysis tasks by extracting latent features of heterogeneous networks. However, in the face of new challenges of increasing network size, diverse latent features, and unseen network noise, existing representation models need to be further optimized. In this paper, a robust and fast representation learning model is proposed for heterogeneous networks, called RFRL. First, the global features of a heterogeneous network are divided into multiple intra-type local features and inter-type local features, and a type-aware biased sampling is designed to generate training samples for each local feature. Second, a node-type-aware and a link-type-aware shallow representation strategy are used to learn intra-type features and inter-type features respectively. This enables the model to achieve good performance while having high speed through the divide-and-conquer learning process and shallow learning model, thus coping with increasing network size and latent feature diversity. Finally, adversarial learning is used to integrate the above two representation strategies to address unseen network noise and enhance the robustness of representation learning. Extensive experiments on three network analysis tasks and three public datasets demonstrate the good performance of our RFRL model.

KEYWORDS

heterogeneous information network (HIN), robust representation learning, adversarial learning, intra-type feature, inter-type feature

1 Introduction

In the real world, many systems (such as traffic systems and social systems) can be abstracted into heterogeneous information networks (HINs) with different node types and link types [1]. However, as HINs grow in size, complex coupled network data cannot cope with the real-time demands of downstream network analysis tasks [2]. For this reason, heterogeneous network representation learning has been proposed and is developing rapidly [3]. Heterogeneous network representation learning is the process of converting high-dimensional complex HINs into low-dimensional simple discrete vectors that retain as much of the underlying features of the network as possible [4]. After heterogeneous network representation learning, the resulting low-dimensional vectors can be used as feature input for downstream network analysis tasks to improve speed and performance [5].

Heterogeneous network representation learning has proved to be very useful in many downstream tasks [6] such as link prediction, node classification, node clustering, etc.

Depending on the depth of the model structure, existing heterogeneous network representation models can be simply classified into shallow and deep models [7]. Shallow models learn the feature representation of nodes by neural networks with fewer layers [8, 9]. In the metapath2vec [10] model, a meta-path guided walk strategy is used to sample node sequences. These node sequences are then used to generate node embedding via a word vector model skip-gram, which maps the feature information of HINs into low-dimensional vectors. The ASPEM [11] model proposes a multi-aspect-based approach to capture the semantic information of HINs by learning low-dimensional discrete vectors of nodes in multiple semantic spaces. Shallow models have fewer model layers and parameters and are therefore faster to train and rarely suffer from overfitting problems [12]. Deep models generally extract feature information in HINs through multiple nonlinear transformations between multiple hidden layers [13]. The HAN model first splits the graph into multiple sub-graphs with the same type guided by meta-paths, and then uses a node-level attention strategy to learn the features of nodes in each sub-graph. Finally, these sub-graphs with different types are mapped into the same feature space by the semantic-level attention strategy. In HetGNN [14], both long short-term memory (LSTM) and Multilayer Perceptron (MLP [15]) are used to extract and understand the feature information of HINs and convert it into low-dimensional discrete vectors. With more complex structures, deeper models can learn higher dimensional features and be applied to more tasks such as image processing, speech recognition, etc. In summary, shallow models are faster than deep models but perform worse.

In the new era of big data, the scale of HINs is getting larger and larger, the network features are getting more and more complex and diverse, and the network noise is getting more and more numerous [16]. Facing the characteristics of HINs in the new era, there are three further challenges to existing models.

- *Speed and performance are difficult to balance.* With the rapid development of IoT [17] and cloud computing technologies, the size of HINs is increasing [18]. Meanwhile, downstream network analysis tasks are increasingly time-sensitive. Existing deep models perform well, but their timeliness cannot meet the demands of the big data era. Existing shallow models have good speed and scalability, but they struggle to accurately capture network features and have yet to improve their performance.
- *Accurate network features are difficult to extract.* In the era of big data, the number of node types and link types in a HIN is also increasing, and the latent features it represents are becoming increasingly complex and diverse. Moreover, multiple features are increasingly coupled and mixed. As a result, feature extraction becomes increasingly difficult. Most traditional models try to extract all features directly and fail to separate multiple features better, thus facing the dilemma of insufficient feature extraction accuracy.
- *The effect of network noise is neglected.* There is often some noise in HINs, such that some nodes are lost and some nonexistent links are constructed. These noises can cause local features of nodes or links to be lost or inaccurate. During model training, this noise increases the generalization error of the model and degrades its performance when dealing with unseen data. Overall, this noise can make the learned heterogeneous network representation vectors less accurate, which can affect the performance of downstream tasks.

To solve the above problems, the motivation of this paper is to design a heterogeneous network representation model that can adapt to network noise with high speed and performance. For one thing, to maintain the high-speed of representation learning, shallow models rather than deep models are chosen to cope with large-scale HINs. And for another, to maintain the performance of representation learning, the features of the heterogeneous network are further decomposed into intra-type and inter-type features. Intra-type features refer to the proximity of multiple nodes under the same node type. Inter-type features refer to the semantic similarity of nodes between two different node types. The accuracy of feature extraction is further improved by converting the original one learning process of global features into multiple learning processes of different sub-features. And thirdly, to attenuate the effect of network noise, generative adversarial networks (GANs [19]) are introduced into the representation learning process to enhance the generalization ability. Based on the above ideas, we propose a robust and fast representation learning model for HINs, called RFRL. The main contributions of this paper are as follows.

- A type-aware bias sampling strategy is proposed to treat each node type and each link type as independent subspaces, and generate both intra-type training samples for each node type and inter-type training samples for each link type using a random walk strategy.
- A node type-aware adversarial learning strategy is designed to learn intra-type features in each node type space using a shallow network, and generate more unseen samples using GAN to enhance the robustness of feature extraction and attenuate the effects of noise.
- A link-type-aware adversarial learning strategy is designed to learn inter-type features in each link type space using another one shallow network, and also to enhance the robustness and generalization of feature extraction using adversarial learning as well.
- The RFRL model is designed to achieve a balance between speed and performance by combining the above strategies. Extensive experiments on three analysis tasks and three public datasets demonstrate the excellent performance of our RFRL model.

The rest of the paper is organized as follows. Related work and definitions are presented in Section 2 and Section 3, respectively. Section 4 shows our RFRL model in detail. The experimental analysis is described in Section 5. Finally, Section 6 concludes the paper.

2 Related work

From the technical perspective, existing models or methods can be simply divided into two categories:

(1) Shallow model-based algorithms. The first shallow representation models are random walk-based models. These models first use a random walk strategy to obtain training samples with both intra-type and inter-type features, and then use a shallow skip-gram model to learn both features simultaneously [10, 20–22]. For example, in metapath2vec [10], the setting of meta-paths guides the model to sample intra-type features and inter-type features. The Spacey [20] model proposes a meta-path-based random walk method for heterogeneous personalized space to collect samples on a meta-graph collapsed from a given meta-path. The HHNE [21] model uses meta-path guided random walk to generate heterogeneous neighborhoods for each node to obtain intra-type features and inter-type features. The MARU [22] model uses a meta-context-aware skip-gram based model to learn dynamic meta-contextual relationships to collect samples. Such algorithms have high speed and performance, but most rely on supervised information given by external experts to guide the learning patterns of intra-type features and inter-type features. The second shallow representation models are decomposition-based models. These models decompose the original network into multiple subnetworks and perform shallow learning for each sub-network [23–25]. For example, the EOE [23] model incorporates a harmonious embedding matrix to further embed the embedding that only encode intra-network links. In RHINE [24], pairs of network links are used to distinguish relations into affiliation relations (ARs) and point-to-point structured interaction relations (IRs) to capture the unique structure of the relations. The MIFHNE [25] method models structural proximity, attribute information, and label information in the framework of non-negative matrix decomposition (NMF). The PME [26] model propose to build object and relation embedding in separate object space and relation spaces. Such algorithms do not rely on external supervised information and are fast in time. However, the integration and fusion of multiple subgraph features is difficult, resulting in the performance of this type of algorithm being weak and stable.

(2) Deep model-based algorithms. To better capture intra-type features and inter-type features, multiple deep models are used to enhance the feature learning capability of the models [27]. For example, in MAGNN [28], inner and outer aggregation of meta-paths are designed to collect samples containing inter-type features and intra-type features. The HAN [29] model proposes a node-level attention mechanism and semantic-level attention mechanism to learn intra-type features and inter-type features of HINs, respectively. Both models are designed to consider intra-type and inter-type features of HINs, but both rely on the setting of meta-paths. The HetSANN [34] model and the HGT model take the same type of node as the center and calculate the importance of other types of nodes around it. These two methods can capture the interactions between different types of nodes well, but do not do specialized learning of intra-type node features. The HetGNN [14] model uses a restarted random walk strategy instead of meta-path-based walk strategy, using multiple artificial neural networks to learn the attributes and structures of the nodes, respectively. The model considers and explicitly uses both artificial neural networks to learn intra-type features and inter-type features. However, due to the complexity of the model structure, the training learning of the model is slow and the generalization ability is not strong [30]. The MV-ACM [31] model is a GAN-based model of multiple views divided by link relations, using a game of generators and discriminators to robustly learn the relations between views.

In summary, shallow models have high speed but relatively weak learning ability for network features; deep models can better capture the nonlinear features of complex networks, but their time complexity is higher. Moreover, existing models rarely consider the effect of noise in the network. With the rapid development of new technologies (such as IoT and cloud computing), HINs are getting more large, heterogeneous, and noisy, and their features are getting more complex. The existing models need to be further improved in the face of new features of HINs. To this end, this paper tries to decompose complex and diverse feature learning into intra-type feature learning for node-type subspaces and inter-type feature learning for link-type subspaces to reduce the learning difficulty and enhance the learning accuracy. Meanwhile, in this paper, we try to design a novel shallow model that guarantees the speed and performance of learning. To reduce the effect of network noise, adversarial learning is incorporated into the shallow model to generate more unseen training samples using adversarial learning, which results in more generalized and robust network features.

3 Definition

In this section, several definitions in this paper are first introduced.

Definition 1: Heterogeneous Information Networks (HINs). The heterogeneous network $G(N, E, T, R)$ is composed of a node set $N=\{n_1, n_2, \ldots, n_n\}$ with node type $T=\{T_1, T_2, \ldots, T_n\}$. and a link set $E=\{e_1, e_2, \ldots, e_n\}$ with link type $R=\{R_1, R_2, \ldots, R_n\}$. The mapping relations between node types T with nodes V and between link types R with links E are φ. Specifically, if nodes v_i and v_j belong to the same node type T_1, then there exists $\varphi(v_i)=\varphi(v_j)=T_1$. The links e_i and e_j belong to the same link type R_1, then there exists $\varphi(e_i)=\varphi(e_j)=R_1$.

Definition 2: Heterogeneous Network Representation Learning. Heterogeneous network representation learning is the process of mapping node v_i to low-dimensional vectors $x_i \in R^{1\times d}$ by learning from a HIN, that is $f(v_i)\rightarrow x_i \in R^{1\times d}$. A feature matrix $X \in R^{|V|\times d}$ is formed with the low-dimensional vectors of all nodes, where $|V|$ is the number of nodes. The feature matrix X of nodes can be used in the analysis of downstream tasks of the network.

4 The proposed model

4.1 Overview

With the advent of the era of big data, complex systems in the real world are getting larger and noisier, and their internal heterogeneity is getting stronger. That is to say, in the network,

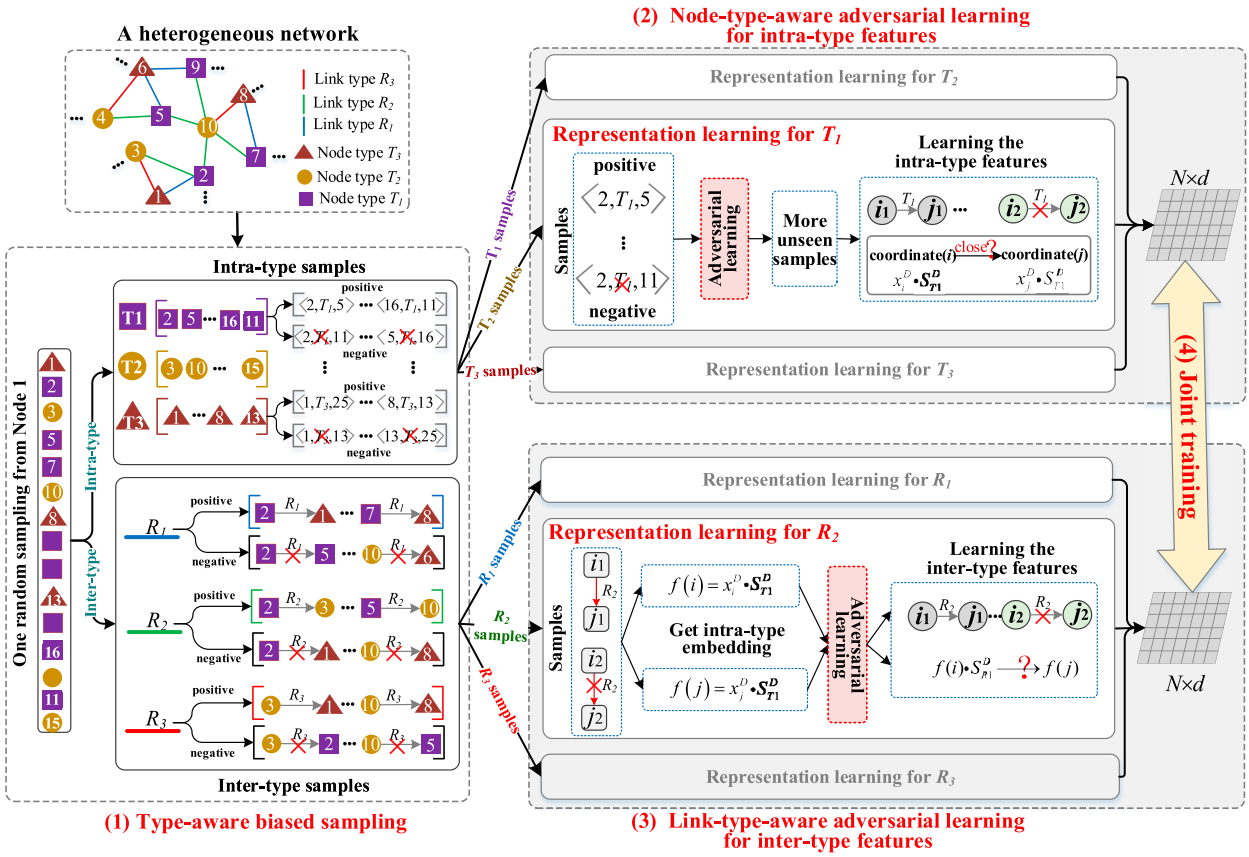

FIGURE 1
The overview of the RFRL model.

the types of nodes and links are increasingly numerous, the features are increasingly complex and diverse, and the correlations between features are increasingly strong. To cope with the change of HINs, the paper constructs a robust and fast heterogeneous network representation model, called RFRL. Based on the idea of "divide and conquer", the model decomposes the heterogeneous network global features into intra-type local features and inter-type local features. Specifically, the model treats each node type and each link type as a feature subspace. The intra-type features refer to node proximity under each node type feature subspace, and inter-type features refer to semantic similarity between each link type feature subspace. Through multiple learning of intra-type and inter-type features instead of one learning of global features, the accuracy of feature learning increased. Moreover, the model uses shallow models instead of deep models to ensure the high-speed of feature learning, and use adversarial learning to enhance generalization of learned features and compatibility with network noise.

The overview of the RFRL model is shown in Figure 1. The whole model contains four parts.

- *Type-aware biased sampling* is the first part. In this part, a type-aware random walk strategy is designed to simultaneously generate intra-type training samples for each node type and inter-type training samples for each link type in one sampling process. Furthermore, the global information of node types is used as weights to generate the final biased intra-type samples of different node types. Meanwhile, the global information of link types is used as weights to generate the final biased inter-type samples of different link types.

- *Node-type-aware adversarial learning* is the second part. In this part, each node type is first viewed as an intra-type feature subspace. Then, based on the idea that "if two nodes are near neighbors, the coordinates of the two nodes in the subspace should be close", a shallow network is designed as a discriminator to accurately learn intra-type features of the subspace. Next, a noisy version of the same shallow model is used as a generator to generate fake intra-type features. The fake features are disguised as more unseen fake samples to cheat the discriminator. Finally, the discriminator identifies real and fake features from real and fake samples to generate more robust features and reduce the influence of network noise.

- *Link-type-aware adversarial learning* is the third part. Similar to above part, each link type is viewed as an inter-type feature subspace. Then, based on the idea that "if a link exists between two nodes with different types, the embedding vector of one node can reach the embedding vector of another node by

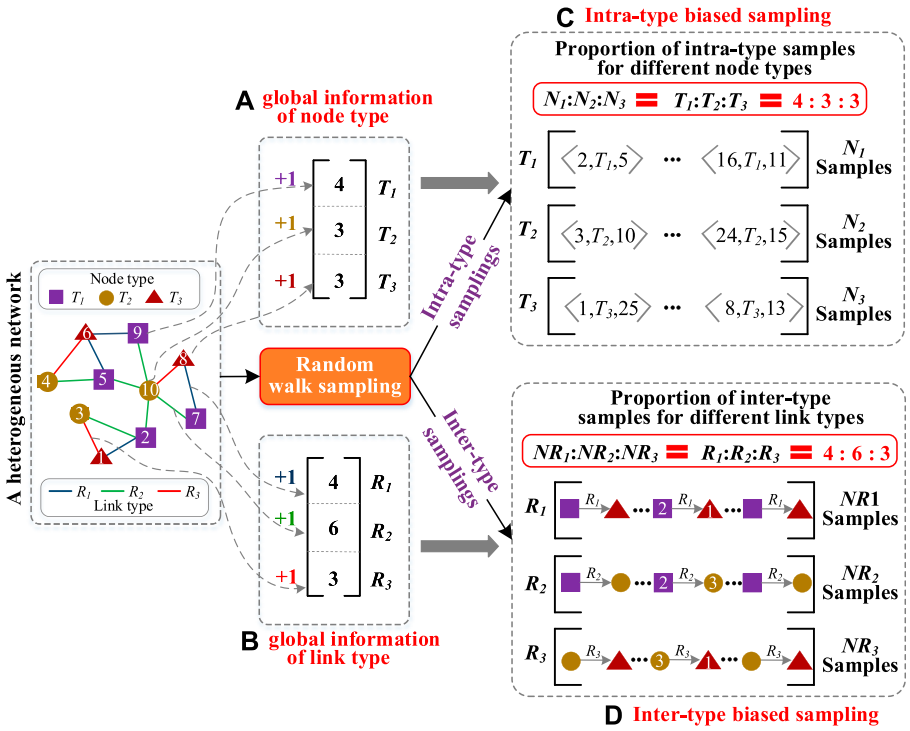

FIGURE 2
The process of type-aware biased sampling. **(A)** calculate global information of node types. **(B)** calculate global information of link types. **(C)** generate intra-type biased samples. **(D)** generate inter-type biased samples.

transforming the semantic features of the link", another shallow network is designed as a discriminator to accurately learn inter-type features of the subspace. Next, a noisy version of the same shallow model is used as a generator to generate fake inter-type features as unseen fake samples to cheat the discriminator. Finally, adversarial learning between generator and discriminator can capture more robust and generalized inter-type features and weaken the impact of network noise.

- *Joint training* is the fourth part. This part uses the intra-type features of each node as shared parameters. By the alternate execution of intra-type feature learning and inter-type feature learning, the global node representation is further improved.

4.2 Type-aware biased sampling

To accurately learn intra-type and inter-type features in HINs, it is essential to generate suitable training samples for these features. This involves generating intra-type training samples for each node type and inter-type training samples for each link type. However, in HINs, different node types have different global distributions (in terms of the number of nodes with different types), and different link types also have different global distributions (in terms of the number of links with different types). This global information determines the importance of different intra-type features or different inter-type features in the global features. Therefore, the intra-type training samples should satisfy the global distribution of node types, and inter-type training samples should satisfy the global distribution of link types. Moreover, due to intra-type features and inter-type features are coupled to each other, it is important to use a same sampling process to generate both intra-type and inter-type training samples, so as to preserve the coupling between them.

Based on the above ideas, supervised by the global distribution information, a type-aware biased sampling strategy is designed to simultaneously generate intra-type samples for each node type and inter-type samples for each link type. The detailed procedure of this strategy consists of three steps, as shown in Figure 2.

Step 1, *calculate global information of node types and link types.* First, grouping the network topology by node type, the proportion of nodes in each group is the weight W_N of all intra-type features in global feature learning, as shown in Figure 2A. Second, grouping the network topology by link type, the proportion of links in each group is the weight W_L of all inter-type features in global feature learning, as shown in Figure 2B.

Step 2, *type-aware sampling.* First, a sample queue $Q_N(T_k)$ is assigned to each node type T_k for storing intra-type feature samples and a sample queue $Q_L(R_s)$ is assigned to each link type R_s for storing inter-type feature samples. After that, a node n_i as n_{pre} is randomly selected from the HIN as the starting point. The node n_{pre} walks randomly from its neighbors to the next node n_{pre} through a link. Following this, node n_{next} as n_{pre} walks to the next node n_{next} at random. Each walked node n_{next} is dropped into the queue Q_N of its type. And the link (n_{pre}, n_{next}) is dropped into the queue Q_L of its type. A walking ends when the number of walking nodes exceeds the preset random walk length. When all nodes in the network are used

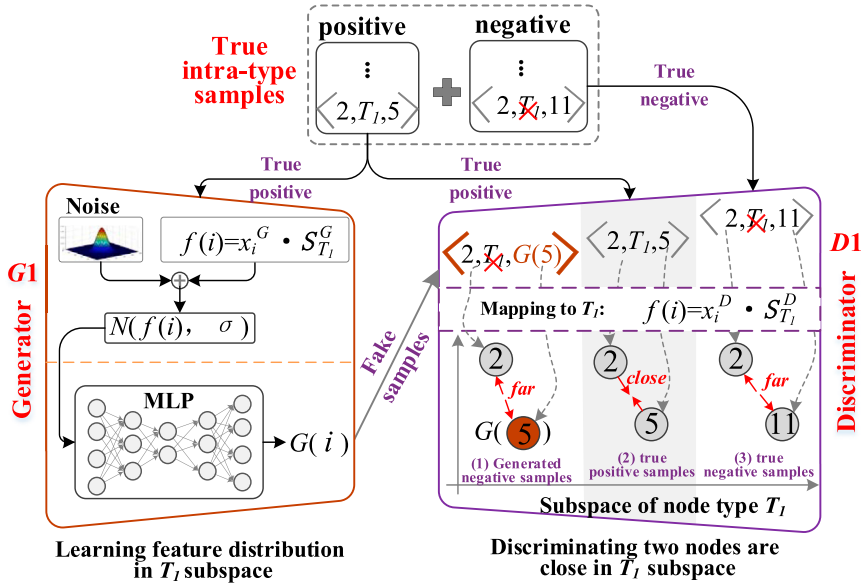

FIGURE 3
The process of node-type-aware adversarial learning.

as the starting point for a walking, the random walk process ends. Finally, in each queue $Q_N(T_k)$, the node sequence is further divided into multiple subsequences with window size L. In each subsequence, any two nodes n_i and n_j can form a positive sample $< n_i, T_k, n_j >$ to preserve 1-order and high-order intra-type features. And, two nodes n_i and n_m in different subsequences can generate a negative sample$< n_i$, no $T_k, n_j >$. Moreover, in each queue $Q_L(R_s)$, a link and its two vertices can generate a positive sample$< n_i, R_s, n_j >$ for the inter-type feature learning of type R_s. And, a positive sample $< n_x, R_l, n_h >$ of type R_l can be transformed to a negative sample $< n_x, R_s, n_h >$of type R_s by replacing R_l with R_s. In the two training samples, the ratio of positive samples to negative samples is 1:3.

Step 3, *generate biased samples*. After sampling, the intra-type training samples (or inter-type training samples) must be proportionally consistent with the weights W_N (or W_L) calculated in step 1. Therefore, the expected number of each intra-type training samples (or inter-type training samples) is first calculated based on the weight W_N (or W_L). Then, for a node type (or link type), if the number of generated samples is larger than the expected number, the redundant part is randomly removed. Finally, if the number of generated samples is smaller than the expected number, the missing parts are randomly copied from the existing samples.

4.3 Node-type-aware adversarial learning for intra-type features

The first important feature of a HIN is the intra-type feature, which refers to the proximity among multiple nodes with the same type. In this paper, each node type corresponds to a feature subspace with a special intra-type feature. Compared to global features, each intra-type feature will be purer, and its learning process will be simpler and accurate. Moreover, the difference in the number of training samples for different node types helps more accurate intra-type features to meet global distribution. In addition, the learning process of multiple intra-type features can be executed in parallel to improve the speed of the representation model. In order to further resist the noise in the network, we try to employ the generative adversarial network (GAN) to learn the distribution of each intra-type feature. This allows us to generate more unseen training samples and reduce the impact of noise on model training. By using the GAN, we are able to capture more robust and generalized intra-type features for nodes.

Based on the above ideas, a node type-aware adversarial learning strategy is designed to use the shallow network instead of deep network to learn more robust and accurate intra-type features of nodes, as shown in Figure 3. The whole strategy consists of two components: the generator and the discriminator, which engage in a game to learn more robust and accurate the intra-type features of nodes. Then, a noised version of the same shallow model is used as a generator to generate fake intra-type features. The fake features are disguised as more unseen fake samples to cheat the discriminator. Next, the discriminator identifies real and fake features from real and fake samples. Finally, adversarial learning between generator and discriminator can capture more robust and generalized inter-type features and weaken the impact of network noise. As an example, the adversarial learning process of the intra-type feature learning for node type T_l is as follows.

4.3.1 Intra-type feature learning

Based on the idea that "if two nodes are neighbors, the coordinates of the two nodes in the subspace should be close", the intra-type feature learning of node type T_l is to project two T_l-type nodes into the same feature subspace, and then pull the

coordinates of the two nodes as close together as possible when the two nodes are neighbors, or push the coordinates of the two nodes as far away as possible when the two nodes are not neighbors. Taking two T_1-type nodes i and j as examples, the intra-type feature learning process is as follows.

First, the two nodes are mapped to the feature subspace of node type T_1, and the position coordinates of the two nodes are obtained respectively, as follows.

$$\begin{aligned} f(i) &= x_i \cdot S^{T_1} \\ f(j) &= x_j \cdot S^{T_1} \end{aligned} \tag{1}$$

Where x_i (or x_j)$\in \mathbb{R}^{1\times d}$ is the global representation vectors of node i (or node j). The $S^{T1}\in \mathbb{R}^{d\times d}$ is the projection matrix of node type T_1, which represents the feature subspace of node type T_1.

Second, since the coordinates of two nodes are vectors, the inner product of the two vectors is used to calculate the proximity of the coordinates of the two nodes in the feature subspace, as follows.

$$dis(i,j) = f(i) \cdot [f(j)]^T \tag{2}$$

Then, the *Sigmoid* function is used to regularize the distance between nodes for easy comparison, as follows.

$$Sim(i,j) = sigmoid(dis(i,j)) = \frac{1}{1+\exp(-dis(i,j))} \tag{3}$$

After regularization, the distance between two nodes is quantified into the range of [0, 1]. If two T_1-type nodes are neighbors, then the distance between the two nodes in the subspace of node type T_1 should converge to 1, otherwise the distance between the two nodes should converge to 0.

4.3.2 Intra-type generator

Based on the above pattern of feature learning, we try to design a noisy feature extractor as a generator **G1** to generate fake intra-type features of node type T_1, so as to help the model tolerate noise and extract more robust and general intra-type features. Specifically, the fake intra-type feature generation process of node type T_1 is as follows.

Fake intra-type feature generation. First, another projection matrix $S_G^{T_1} \in \mathbb{R}^{d\times d}$ of node type T_1 is generated and randomly initialized, which represents the feature subspace of node type T_1 in the generator **G1** and competes with the projection matrix $S^{T1}$of the T_1-type real feature subspace.

Second, one node i is mapped to the T_1-type fake feature subspace of **G1**, and the position coordinate is calculated respectively, as follows.

$$f_G(i) = x_i \cdot S_G^{T_1} \tag{4}$$

Where $f_G(i)\in \mathbb{R}^{1\times d}$ is the coordinate of node i in T_1-type real feature subspace.

Then, to generate noisy fake intra-type features, we use Gaussian noise to disturb the coordinate of node i in the T_1-type fake feature space of **G1**, defined as follows.

$$f_G^{noise}(i) = N(f_G(i), \sigma^2 I) \tag{5}$$

Where the σ is the variance of Gaussian noise, which is a preset hyper-parameter. The I is the unit vector, $N(*,\sigma)$ is a function to generate a Gaussian distribution with mean * and variance σ.

Finally, we use a multi-layer perceptron to enhance the nonlinearity of the noise coordinate of node i, and defined as.

$$f_G^{noise}(i) = g(\cdots g(f_G^{noise}(i)\cdot W_1 + b_1)\cdots W_k + b_k) \tag{6}$$

Where $W\in \mathbb{R}^{d\times d}$ is the parameter matrix of the MLP and $b\in R^{1\times d}$ is the bias of the MLP. And the MLP is set to one layer in this paper. The $g()$ is the nonlinear activation function(*LeakyReLU* is used in this paper).

Loss function. To ensure the effectiveness of the generated fake feature distribution, we hope that the generated fake feature distribution is as close as possible to the real distribution. To achieve this goal, we use the positive intra-type samples <*i*, T_1, *j*> of node type T_1 to train the generator **G1**.

Therefore, the loss function of generator **G1** consists of two parts. The first part is that the coordinates of two nodes of one positive sample in **G1** should be as close as possible, defined as follows.

$$Loss_1 = -\sum_{<i,j>\in \left\{ \begin{smallmatrix} postive \\ samples \end{smallmatrix} \right\}} \log(Sim(f_G^{noise}(i), f_G^{noise}(j))) \tag{7}$$

The second part is that the fake coordinate in **G1** of any node in one positive sample should be as close as possible to the coordinate in the real subspace, defined as follows.

$$\begin{aligned} Loss_2 = -\sum_{<i,j>\in \left\{ \begin{smallmatrix} postive \\ samples \end{smallmatrix} \right\}} & \log(Sim(f_G^{noise}(i), f(i))) \\ & + \log(Sim(f_G^{noise}(j), f(j))) \end{aligned} \tag{8}$$

Where $f(j)$ and $f(i)$ is the representation vector (the coordinate) in the real feature subspace of node type T_1.

The final loss of generator **G1** is defined as follows.

$$Loss_{G1} = Loss_1 + Loss_2 \tag{9}$$

4.3.3 Intra-type discriminator

As a game competitor of the generator **G1**, a discriminator **D1** needs to be constructed. In this way, the generator **G1** and the discriminator **D1** form an adversarial generative network GAN. In this paper, to simplify the structure of GAN, the **D1** needs to have two capabilities. The first capability is to learn the true feature distribution in the feature subspace of node type T_1. The second capability is to identify the real feature distribution and the fake feature distribution.

Real intra-type feature learning and discrimination. The fake feature distribution f_G^{noise} generated by **G1**, positive intra-type samples <*i*, T_1, *j*> and negative intra-type samples<*i*, *not* T_1, *j*> generated by the type-aware biased sampling strategy are inputs of the discriminator **D1**. To achieve the two capabilities of **D1**, three inputs are transformed into three types of training samples.

The first type of training samples are real positive samples <*i*, T_1, *j*>$\in$*RPS*, and they are generated by the type-aware biased sampling strategy. For a real positive sample<*i*, T_1, *j*>, the proximity of the coordinates of two nodes *i* and *j* in the T_1-type true feature subspace of should be 1. That is to say, $Sim(i,j) = 1$ according to Equation 3.

The second type of training samples are real negative samples <*i*, *not* T_1, *j*>$\in$*RNS*, and they are also generated by the type-aware biased

sampling strategy. For a real negative sample<*i, not* T_1, *j*>, the proximity of the coordinates of two nodes *i* and *j* in the T_1-type true feature subspace of should be 0. That is to say, $Sim(i,j) = 0$ according to Equation 3.

The third type of training samples are fake positive samples <*i*, T_1, $G(j)$>(or <$G(i)$, T_1, *j*>)$\in FPS$ where the $G(j)$ represents the fake feature vector of node *j* in **G1**. For a fake positive sample <*i*, T_1, $G(j)$ >, the proximity of the coordinates of two nodes *i* and *j* in the T_1-type true feature subspace should be 0. That is to say, $Sim(i,j) = 0$ according to Equation 3.

When the discriminator **D1** is trained only with real positive and real negative samples, the **D1** can accurately capture the true feature distribution in the T_1-type feature subspace. When the discriminator **D1** is trained with real positive and fake positive samples, the **D1** can accurately identify the real and fake feature distribution. Therefore, through the combined training of three training samples, the **D1** can not only learn the real feature distribution, but also distinguish the real and fake feature distributions. Moreover, driven by the fake feature distribution generated by generator **G1**, a large number of unseen training samples are generated. These fake samples can further help **D1** to learn more robust and generalized intra-type features, and reduce the influence of network noise.

Loss function. According to three types of training samples, the loss function of discriminator **D1** consists of three parts, defined as.

$$\begin{aligned} Loss_{D1} = & \sum_{<i,j>\in RPS} -\log\left(Sim\left(f(i), f(j)\right)\right) \\ & + \sum_{<i,j>\in RNS} -\log\left(1 - Sim\left(f(i), f(j)\right)\right) \\ & + \sum_{<i,j>\in FPS} -\log\left(1 - Sim\left(f(i), f_G^{noise}(j)\right)\right) \\ & - \log\left(1 - Sim\left(f_G^{noise}(i), f(j)\right)\right) \end{aligned} \tag{10}$$

4.4 Link-type-aware adversarial learning for inter-type features

The second important feature of a HIN is the inter-type feature, which refers to the semantic similarity between two node types. That is to say, if there is a link of type R_1 between node *i* of type T_1 and node *j* of type T_2, then the node *i* can reach the node *j* through the semantic relation R_1. In this paper, each link type is also regarded as a feature subspace possessing a unique inter-type feature that enables the semantic transformation of two heterogeneous nodes. More specifically, if a link of type R_1 exists between node *i* of type T_1 and node *j* of type T_2, then the T_1-type intra-type feature of node *i* can be similar to the T_2-type intra-type feature of node *j* through the semantic transformation of relation R_1, abbreviated as <*i*, T_1, R_1, T_2, *j*>. Similar to the intra-type feature learning, the difference in the number of training samples for different link types helps more accurate inter-type features to meet global distribution. To reduce the impact of network noise, we also employ the generative adversarial network (GAN) to learn the more robust and generated inter-type feature distribution.

Based on the above ideas, a link-type-aware adversarial learning strategy is designed to learn more robust and accurate inter-type features of nodes, as shown in Figure 4. The whole strategy consists of generators and discriminators, each of which uses a shallow network. It is important to note that the inter-type feature learning aims to capture the relation between two intra-type feature spaces. For instance, in the case of link type R_1, we use a generator and a discriminator to learn inter-type features as follows.

4.4.1 Inter-type feature learning

Based on the idea that "if a link exists between two nodes with different types, the embedding vector of one node can be transformed to the embedding vector of another node by the semantic features of the link", the inter-type feature of link type R_1 is to learn the transform from the intra-type feature of one node to the intra-type feature of another node in the R_1 feature subspace. Then, the two intra-type feature vectors are pushed close when there is an R_1 relation between them, and the two feature vectors are drawn far apart when there is not an R_1 relation between them. Taking T_1-type node *i* and T_2-type node *j* with an edge of type R_1 as example<*i*, T_1, R_1, T_2, *j*>, the inter-type feature leaning process is as follows.

First, the intra-type feature $f(i)$ of node *i* in T_1-type feature space and the intra-type feature $f(j)$ of node *j* in T_2-type feature space are first obtained from the Section 4.3. Meanwhile, the link type R_1 is regarded as a semantic feature subspace, and a transformation matrix S_{R1} is defined to represent the semantic transformation process in the subspace.

Second, the intra-type feature $f(i)$ of T_1-type node *i* is mapped to the R_1 feature subspace, and defined as.

$$Tran(i) = f(i) \cdot S_{R_1} \tag{11}$$

Where $S_{R1} \in \mathbb{R}^{d\times d}$ is the semantic transformation matrix of link type $_{R1}$, and the $\cdot$ is the function of matrix multiplication.

Then, the similarity between the transformation feature $Tran(i)$ of node i and the intra-type features $f(j)$ of node *j* is calculated as.

$$Sim(i,j) = Tran(i) \cdot \left[f(j)\right]^T \tag{12}$$

Finally, the *Sigmoid* function is used to regularize the similarity to the range of [0,1], as follows.

$$Sim(i,j) = \frac{1}{1+\exp\left(-Sim(i,j)\right)} = \frac{1}{1+\exp\left(-Tran(i)\cdot\left[f(j)\right]^T\right)} \tag{13}$$

The similarity is adjusted according to the correctness of the semantic transformation of R_1. Specifically, the value of similarity should be increased to 1 when $Tran(i)$ can correctly reach the T_2 features space. Conversely, the value of similarity should be decreased to 0.

4.4.2 Inter-type feature generator

Same as the intra-type generator, the generator **G2** is used to generate the noisy fake inter-type features and apply them to the inter-type discriminator **D2**, so as to learn more robust features and reduce the impact of the network noise. The generating process for the fake inter-type features of R_1 is as follows.

Fake inter-type feature generation. First, another projection matrix $S_G^{R_1} \in \mathbb{R}^{d\times d}$ of link type R_1 is generated and randomly initialized, which represents the fake feature subspace of R_1 in

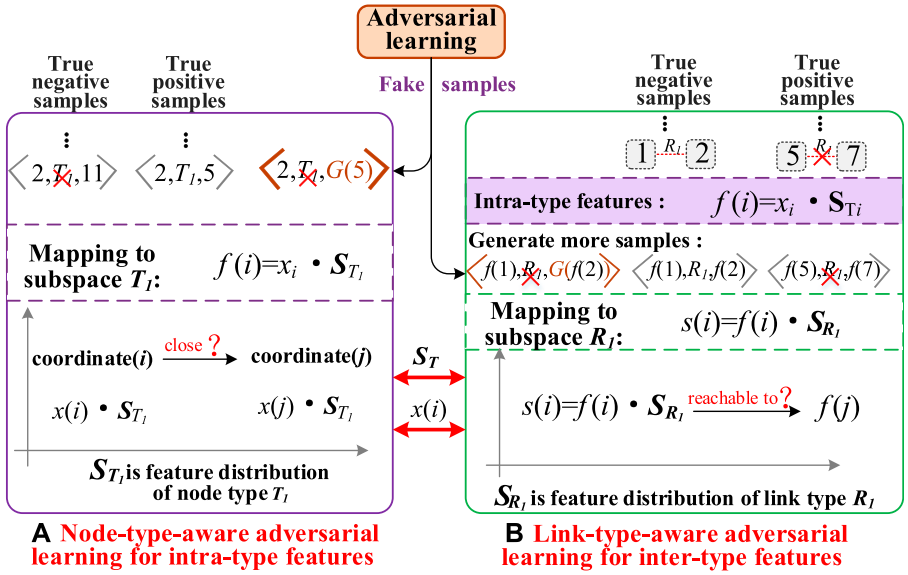

FIGURE 4
Comparison of intra-type feature learning and inter-type feature learning. **(A)** Node-type-aware adversarial learning for intra-type features. **(B)** Link-type-aware adversarial learning for intra-type features.

G2. And, the intra-type feature of node *i* is transformed by the fake feature subspace in **G2** to obtain the transformation feature $Tran_G(i)$.

$$Tran_G(i) = f_G(i) \cdot S_{R1}^G \tag{14}$$

Second, to better obtain the fake inter-type feature, we add some Gaussian noise to the inter-type feature $Tran_G(i)$, defined as follows.

$$Tran_G^{noise}(i) = N(Tran_G(i), \sigma^2 I) \tag{15}$$

Where the size of the variance σ can be adjusted to control the degree of disturbance of Gaussian noise.

Finally, to enhance the expressiveness of $Tran_G^{noise}(i)$, the model continues using the MLP model with two layers to obtain the final fake inter-type features in the following way.

$$Tran_G^{noise}(i) = g(\cdots g(Tran_G^{noise}(i) \cdot W_1 + b_1) \cdots W_k + b_k) \tag{16}$$

Where $W \in \mathbb{R}^{d \times d}$ is the parameter matrix of the MLP and $b \in R^{1 \times d}$ is the bias of the MLP. And the $g()$ is the nonlinear activation function (*LeakyReLU* is used in this paper).

Loss function: For inter-type features, the concern is whether the node intra-type features can reach the correct feature subspace after the semantic transformation. Therefore, we use the *cross-entropy* loss to make the fake inter-type features *Trannoise* $G(i)$ similar to the true intra-type features $f(j)$, so that the fake inter-type features of type T_1 can arrive correctly in the feature subspace of T_2. To achieve this goal, we use the positive inter-type samples $<i, T_1, R_1, T_2, j>$ to train the generator **G2**. The details are as follows.

$$Loss_{G2} = -\sum_{<i,j> \in \left\{ \begin{matrix} postive \\ samples \end{matrix} \right\}} \log(Sim(Tran_G^{noise}(i), f(j))) \tag{17}$$

4.4.3 Inter-type feature discriminator

Similarly, we need an inter-type discriminator **D2** as a competitor to the inter-type generator **G2**. In this way, the generator **G2** and the discriminator **D2** form an adversarial generative network GAN. Similar to **D1**, **D2** also needs to have two capabilities. The first capability is to learn real inter-type features. The second capability is to achieve discrimination between true inter-type features and fake inter-type features.

Real inter-type feature learning and discrimination. The fake feature distribution *Trannoise* $G(i)$ generated by **G2**, positive inter-type samples $<i, T_1, R_1, T_2, j>$ and negative intra-type samples $<i, T_1,$ not $R_1, T_2, j>$ generated by the sampling strategy are inputs of the discriminator ***D2***. To achieve the two capabilities of *D2*, three inputs are transformed into three types of training samples.

The first type of training samples are real positive samples. For the real positive samples $<i, T_1, R_1, T_2, j> \in RPS$, the similarity of the inter-type features $Tran(i)$ and $f(j)$ should be 1. That is to say, $Sim(i,j) = 1$ according to Equation 13.

The second type of training samples are real negative samples. For a real negative sample $<i, T_1,$ not $R_1, T_2, j> \in RNS$, the similarity of the inter-type features $Tran(i)$ and $f(j)$ should be 0. That is to say, $Sim(i,j) = 0$ according to Equation 13.

The third type of training samples are fake positive samples $<G2(i), T_1, R_1, T_2, j>$ (or $<i, T_1, R_1, T_2, G2(j)> \in FPS$. For a fake positive sample $<G2(i), T_1, R_1, T_2, j>$, the similarity of the inter-type features $Tran_G^{noise}(i)$ and $f(j)$ should be 0. That is to say, $Sim(G2(i), j) = 0$ according to Equation 13.

When discriminator **D2** is trained with only true positive samples and true negative samples, **D2** can accurately capture the true inter-type feature distribution. When discriminator **D2** is trained with true positive and fake positive samples,

D2 can accurately identify the true and fake feature distributions. Therefore, through the joint training of three training samples, **D2** can not only incrementally learn the true inter-type features, but also achieve the discrimination of true inter-type features and fake inter-type features. In addition, the large number of unseen training samples generated by generator **G2** can further help *D*2 to achieve robust and accurate learning of inter-type features and thus reduce the effect of network noise.

Loss function. Based on the above idea, the *cross-entropy* loss function is used as the loss of *D*2, and defined as follows.

$$\begin{aligned} Loss_{D2} = & \sum_{<i,j>\in RPS} -\log\left(Sim\left(Tran(i), f(j)\right)\right) \\ & + \sum_{<i,j>\in RNS} -\log\left(1 - Sim\left(Tran(i), f(j)\right)\right) \\ & + \sum_{<i,j>\in FPS} -\log\left(1 - Sim\left(Tran_G^{noise}(i), Tran(i)\right)\right) \end{aligned} \quad (18)$$

4.5 Joint learning

Comparing intra-type feature learning with inter-type feature learning, it is not difficult to get the following finding, as shown in Figure 4. First, the two learning processes use different shallow learning strategies. Intra-type feature learning attempts to learn the positional proximity of the projected coordinates of two nodes in the same subspace. Inter-type feature learning attempts to learn the semantic similarity between two nodes by the semantic transformation of one link type. Second, the two learning processes are closely related to each other. The global representation vector of each node is shared in the two learning processes. Moreover, the learned intra-type features of node types are used as input in the inter-type feature learning process. In addition, an adversarial learning strategy is used in two learning processes to capture more robust and generalized network features and reduce the impact of network noise.

Therefore, node-type-aware intra-type feature learning and link-type-aware inter-type feature learning need to be jointly trained for better performance. In our experiment, the intra-type feature learning process is performed for 5 consecutive epochs with separate sampling, where the sampled intra-type samples are used and the inter-type samples are kept. After the intra-type features are learned, the inter-type feature learning process are performed without sampling for 5 epochs, where each epoch uses the inter-type samples reserved by the intra-type feature learning process. This can help the two learning processes capture the coupling between the intra-type feature and inter-type feature. All training procedures use the stochastic gradient descent algorithm (SGD) for parameter updates. The ratio of the iterations of generator and discriminator is adjusted during training to balance the learning rates of both, resulting in a steady improvement in the performance of both. In addition, adjusting the ratio of two learning rates also controls the learning speed of both. However, it is important to note that adjusting the learning rate may cause the performance of the generator and the discriminator to degrade. In this paper, the learning rate of the generator and discriminator is set to 1e-4.

5 Experiments

5.1 Experimental setup

Datasets. We select three datasets with different sparsity levels, which are related to the literature citation network, the shopping network, and the business network. (1) *DBLP network* is a citation network with journal and conference bibliographic information (https://dblp.uni-trier.de/xml/). (2) *Amazon network* comes from the user and product information of the Amazon platform (http://jmcauley.ucsd.edu/data/amazon/). (3) *Yelp network* contains information on merchants and users in multiple cities in the United States (https://www.yelp.com/dataset). The details of each dataset are shown in Table 1.

Baselines. In the comparison experiments, we selected five models as baselines: two shallow models (RHINE [24] and Metapath2vec [10]), two deep models (HAN [29] and HGT [32]), and one GAN-based representation model (HeGAN [33]). In addition, recommended meta-paths and default parameter settings are used for all models. The details of the baselines are shown in Table 2.

Tasks and metrics. We chose the following three tasks and five metrics to comprehensively evaluate the performance of our model. (1) *Node classification*. Based on the learned node representation vector, the classifier predicts the labels of the nodes. In this task, we use Macro_F1 and Micro_F1 metrics to evaluate the performance of the node classification task. (2) *Node clustering*. Based on the learned node representation vector, the nodes are divided into multiple clusters, where each cluster represents a category. We evaluate the performance of the node clustering task using the NMI metric. (3) *Link prediction*. Based on the learned node representation vector, we predict whether there is a link between two nodes. We use AUC and ACC metrics to evaluate the performance of link prediction tasks.

Setting. The experimental platform is a PC server equipped with an NVIDIA T4 card. The server is outfitted with a 32-core Intel Xeon Cascade Lake (2.5 Hz) processor, 64 GB of RAM, and the Ubuntu 18.04 operating system. The RFRL algorithm was programmed using the PyCharm IDE.

5.2 Node classification

In this section, we perform the node classification task on three datasets Amazon, DBLP, and Yelp, compared to five baselines to test our performance. In this experiment, to verify the stability, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% of the real labeled samples as 9 training datasets are used to train all models, and 10% of other real labeled samples are used as the test set. For the classifier, we use *SoftMax* to calculate the probability of each category and evaluate the classification result with Macro_F1 and Micro_F1 metrics.

Figure 5 shows the accuracy of node classification on 3 different networks, where Figures 5A–C are the Macro_F1 metrics and Figures 5D–F are the Micro_F1 metrics. In

TABLE 1 Datasets.

Dataset	Nodes	Number of nodes	Relation	Number of relations	Avg.Degree
DBLP	Author(A)	14,475			9.04
	Paper(P)	14,376	P-A	41,794	
	Conference(C)	20	P-C	14,376	
	Type(T)	8,920	P-T	114,624	
Amazon	User(U)	344			10.29
	Item(I)	95	I-U	365	
	View(V)	3,773	I-V	195,791	
	Brand(B)	5	I-B	95	
Yelp	User(U)	1,286			19.77
	Business(B)	2,614	B-U	30,838	
	Service(S)	2	B-S	2,614	
	Star(St)	9	B-St	2,614	
	Reservation(R)	2	B-R	2,614	

TABLE 2 Baselines.

Algorithms	Full name	Implement
RHINE [24]	Relation Structure-Aware Heterogeneous Information Network Embedding	Python
Metapath2vec [10]	metapath2vec: Scalable Representation Learning for Heterogeneous Networks	Python
HAN [29]	Heterogeneous Graph Attention Network	Python
HGT [32]	Heterogeneous Graph Transformer	Python
HeGAN [33]	Adversarial Learning on Heterogeneous Information Networks	Python

each graph, we used different colored lines to indicate the performance scores of different algorithms. By observing, we can obtain the following results. (1) *Comparing all models*, the best average performance in the three datasets is RFRL and HGT, followed by HeGAN, HAN, and finally RHINE, metapath2vec. For example, in the Macro_F1 metric, RFRL outperforms the second-best HGT by 0.76% on average across the three datasets; In the Micro_F1 metric, the average performance of RFRL is 1.01% higher than that of the second-best HGT. (2) *Considering the stability*, the stability of the RFRL model is better than that of the shallow model and better than that of the deep model on three networks. For example, on Macro_F1 of the DBLP dataset, the performance drift range of 3.75% for RFRL is lower than the ranges of 4.33% for metapath2vec and 5.21% for HGT. Meanwhile, for the training set with fewer labels, the RFRL model outperforms the other models. For example, on Amazon network, the Micro_F1 score of our RFRL model is 6.22% higher than the second-best model in the training set with 20% labels. (3) *Comparing the structure of the models*, the deep models (HGT and HAN) generally outperforms the shallow models (metapath2vec and RHINE). For example, on the Amazon dataset, the average performance of Macro_F1 of HGT is 3.01% higher than that of RHINE. (4) *Considering the potential of capturing information*, the RFRL model has a better potential to capture information than other models. That is to say, it can obtain good performance on fewer training sets. For example, on the Amazon training sets with 10%–30% labels, RFRL is on average about 3% better than the second-best Macro_F1 metric. (5) *In summary*, the RFRL model has high generalization ability and performance, especially when most of the node information is unknown (few label samples).

5.3 Node clustering

In this section, we test our performance on the node clustering task by choosing the same datasets and baselines as in the above experiments. In this experiment, we use the k-means algorithm to divide the nodes into multiple clusters based on the learned node representation vectors, each of which is a category. Finally, the NMI metric is used to evaluate the consistency of these categories with the true labeled categories.

Table 3 shows the comparison of the best clustering results of the six algorithms on the three datasets. The following

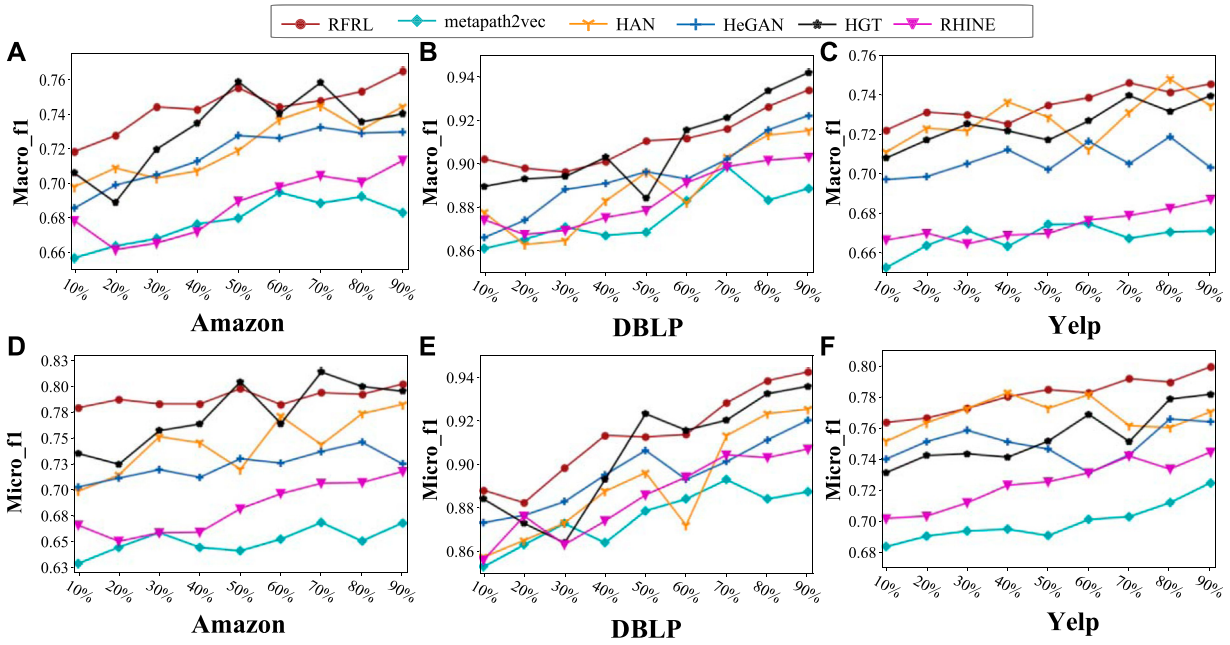

FIGURE 5
Performance of multiple algorithms in node classification. **(A)** The Macro-f1 scores of multiple algorithms on the Amazon dataset; **(B)** The Macro-f1 scores of multiple algorithms on the DBLP dataset; **(C)** The Macro-f1 scores of multiple algorithms on the Yelp dataset; **(D)** The Micro-f1 scores of multiple algorithms on the Amazon dataset; **(E)** The Micro-f1 scores of multiple algorithms on the DBLP dataset; **(F)** The Micro-f1 scores of multiple algorithms on the Yelp dataset.

TABLE 3 Performance of multiple algorithms in node clustering.

	Amazon	Yelp	DBLP
	NMI	NMI	NMI
metapath2vec	0.2989	0.3069	0.6738
RHINE	0.3479	0.3739	0.7352
HAN	0.3893	0.3631	0.7831
HGT	0.3981	0.3871	0.7438
HeGAN	0.367	0.3965	0.7689
RFRL	0.4375	0.4253	0.7576

conclusions are drawn from Table 3. (1) *Comparing all models*, the clustering performance of RFRL is higher than baseline algorithms in the three datasets, which proves the effectiveness and accuracy of RFRL. (2) *Considering the stability*, RFRL is more stable relative to the other models. For example, the smallest differences from the peak in the three datasets are 0.802% for RFRL, 2.75% for HeGAN, and 3.68% for HAN. (3) *Comparing the structure of the models*, GAN-based models (HeGAN, RFRL) generally outperform the other models. For example, in the Yelp dataset, the NMI of HeGAN is 0.3965 and that of RFRL is 0.4253; while the NMI of the other models is below 0.3872. In addition, the deep models (HAN, HGT) generally outperform the shallow models (metapath2vec, RHINE). For example, the NMI of HAN and HGT are roughly 1%–9% higher than metapath2vec and RHINE in all three datasets. (4) *Comparing GAN-based models*, RFRL outperforms HeGAN overall. For example, in the Amazon dataset, HeGAN has an NMI of 0.397 and RFRL has an NMI of 0.4375. (5) *In summary*, the RFRL model is robust and effective in the node clustering task.

5.4 Link prediction

In this section, we test our model using the link prediction task. In this task, we concatenate the node representation vector at the two ends of the link as a low-dimensional feature vector of the link, and then use logistic regression to implement a binary classification to determine the existence of a link. As with node classification, to demonstrate the stability of the model, we still use 5 baselines as competitors and 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% of the real labeled samples as 9 training datasets for comparison experiments.

Figure 6 shows the link prediction performance of the six algorithms on three datasets. The following conclusions can be drawn from the observations. (1) *Comparing the ACC of all models*, the best average performance is achieved by RFRL and HeGAN, followed by HGT and HAN, and finally RHINE and metapath2vec. For example, in the ACC metrics, the average ACC performance of RFRL is 66.53%, and the average ACC performance of HAN is 64.97%. (2) *Comparing the AUC of all models*, the performance of HeGAN has decreased. The best performances on average were RFRL and HGT, followed by

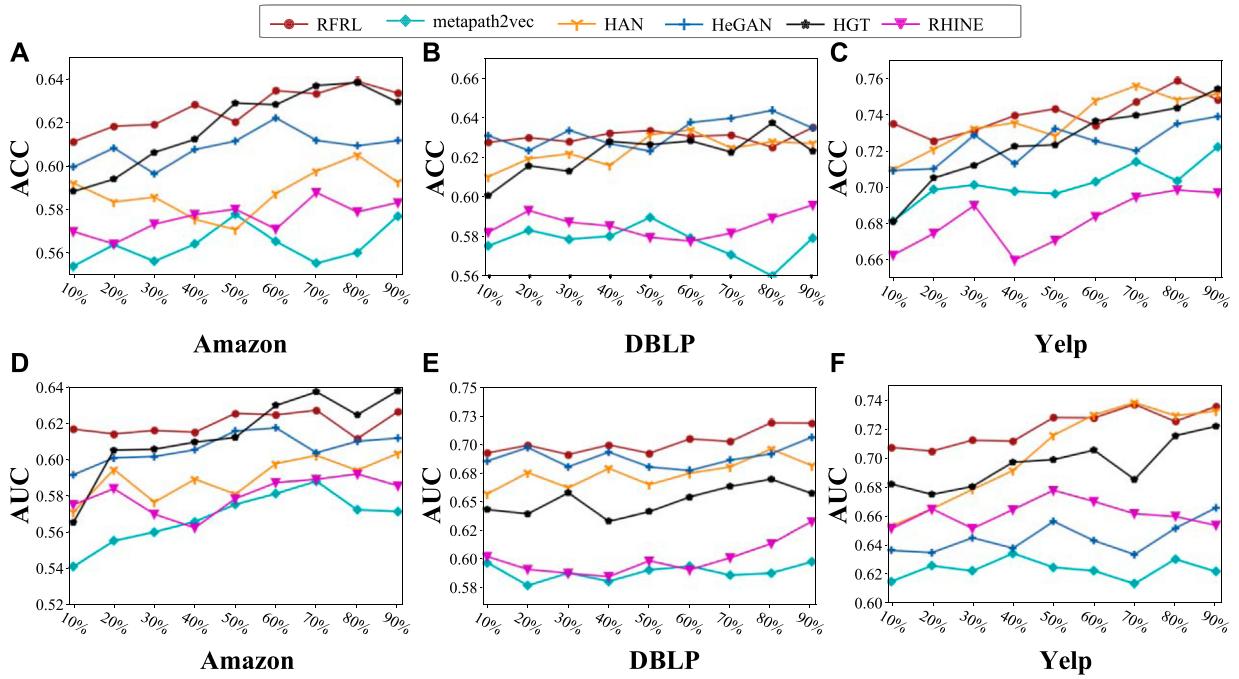

FIGURE 6
Performance of multiple algorithms in link prediction. **(A)** The ACC scores of multiple algorithms on the Amazon dataset; **(B)** The ACC scores of multiple algorithms on the DBLP dataset; **(C)** The ACC scores of multiple algorithms on the Yelp dataset; **(D)** The AUC scores of multiple algorithms on the Amazon dataset; **(E)** The AUC scores of multiple algorithms on the DBLP dataset; **(F)** The AUC scores of multiple algorithms on the Yelp dataset.

HeGAN and HAN, and finally RHINE and metapath2vec. For example, on the Yelp dataset, the average performance of HeGAN is 64.50% and that of RFRL is 72.10%, a difference of 7.60%. (3) *Considering the stability*, the GAN-based models RFRL and HeGAN are the best. For example, in the DBLP dataset, the extreme differences of RFRL and HeGAN are 2.76% and 2.89%. Meanwhile, the performance of these two GAN-based models is relatively better in the training set with 30% labels. For example, on the ACC of Amazon dataset, the average performance of RFRL and HeGAN is 61.62% and 60.15%. (4) *Comparing the structure of the models*, the GAN-based models (RFRL and HeGAN) outperform the other models on average, and the deep models (HGT and HAN) outperform the shallow models (RHINE and Metapath2vec) overall. For example, in DBLP dataset, the average AUC value of HGT is 4.41% higher than that of metapath2vec and 3.57% higher than that of RHINE. (5) *In summary*, RFRL outperforms the other baselines by 1%–13% on the link prediction task. The good performance of RFRL on the datasets with less real samples demonstrates the effectiveness and robustness of the model.

5.5 Additional experiments

To further demonstrate the advantages of the biased sampling strategy, the robustness and Scalability of the model, we perform the following additional comparison experiments based on the link prediction task.

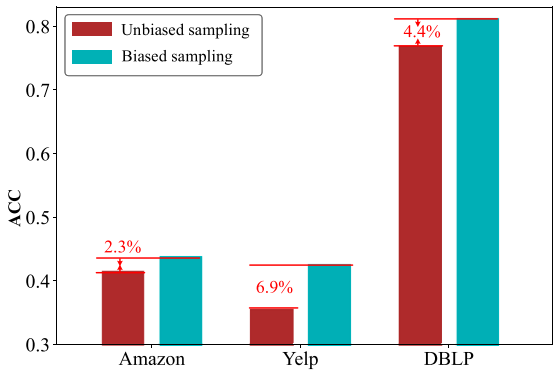

FIGURE 7
Performance of biased and unbiased sampling strategy in link prediction.

5.5.1 The biased sampling strategy

In this experiment, the samples collected by biased and unbiased sampling strategy are fed into our RFFL model as two different models for training, and finally, the performance of the two models is evaluated on the link prediction task using the ACC metric.

Figure 7 shows the performance on the ACC metric for link prediction with two different sampling strategies. In the figure, red bars represent the results of the unbiased sampling strategy and blue bars represent the results of the biased sampling strategy. We can get the following findings. (1) *Comparing*

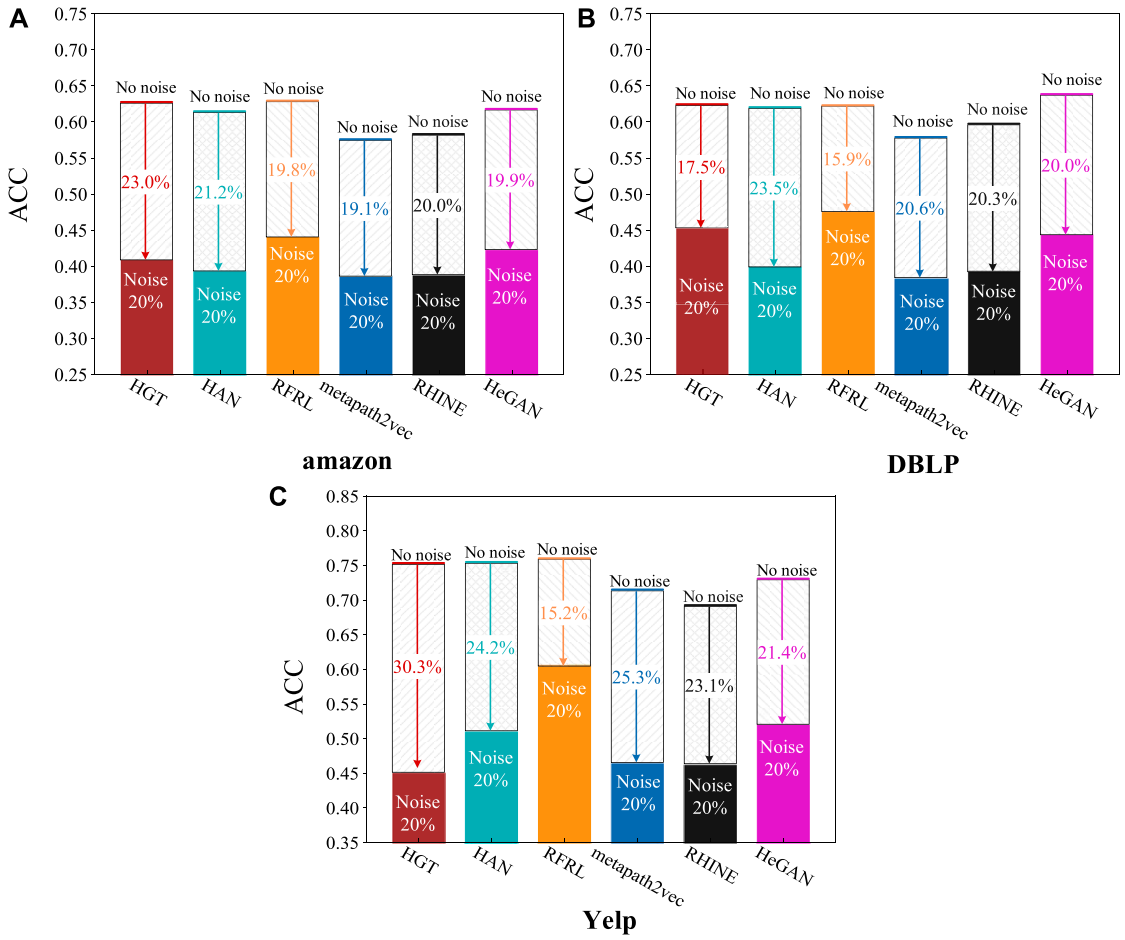

FIGURE 8
Performance of multiple algorithms in the noise networks. **(A)** The ACC scores of multiple algorithms on the Amazon dataset with noise; **(B)** The ACC scores of multiple algorithms on the DBLP dataset with noise; **(C)** The ACC scores of multiple algorithms on the Yelp dataset with noise.

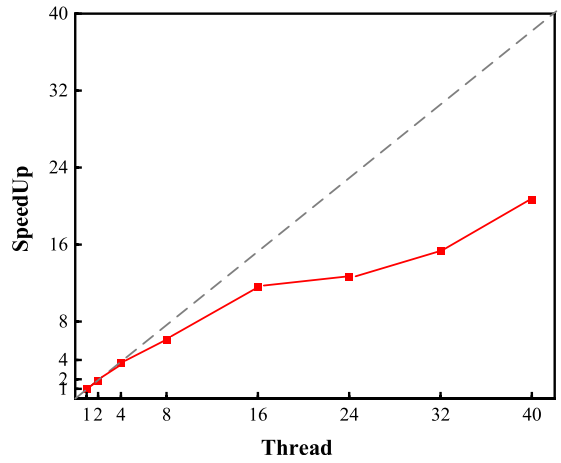

FIGURE 9
The scalability performance of the RFRL model.

the two strategies, the overall performance of biased sampling is better than random sampling. For example, on the Amazon, Yelp and DBLP datasets, the performance improves by 2.25%, 6.93% and 4.4%, respectively. This demonstrates that the biased sampling can indeed better preserve the feature information of heterogeneous networks. (2) *Comparing different datasets*, the performance optimizations of the biased sampling strategy on different datasets are different. For example, the optimization on the Yelp dataset is 6.93%, while on the DBLP it is only 2.25%. This may be related to the sparsity of the data, as well as the number of types. Specifically, the Yelp dataset has a larger number of types and degrees than the other two datasets. (3) *In summary*, the biased sampling strategy captures the intra-type and inter-type features in the network better and performs better.

5.5.2 Robustness

In this experiment, a noise network is obtained by randomly removing 20% of the links from the original heterogeneous network. On the noise network, RFRL is compared with other baseline algorithms in terms of ACC metric for link prediction. The ratio of the training set to the test set is 9:1.

Figure 8 shows the link prediction performance of different algorithms on the noise networks. From the figure, we can

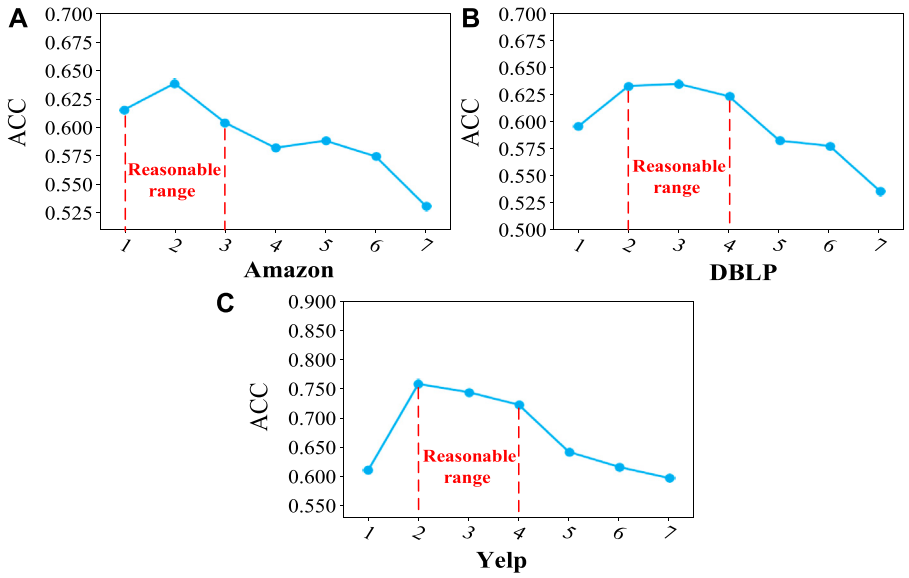

FIGURE 10
Sensitivity analysis of the hyper-parameter *L*. **(A)** The ACC scores of the different parameters L on the Amazon dataset; **(B)** The ACC scores of the different parameters L on the DBLP dataset; **(C)** The ACC scores of the different parameters L on the Yelp dataset.

observe the following information. (1) *Comparing all the models*, the overall performance of all models decreased. RFRL showed the smallest decrease in performance with an average decrease of 16.96%; followed by HeGAN, RHINE, metapath2vec with average decreases of 20.43%, 21.13%, 23.6%, and HAN, HGT with average decreases of 22.97% and 23.60%, respectively. (2) *Comparing different datasets*, the model performance degradation varies, which is related to the structure and semantics of the network. Although the degradation of RFRL in the Amazon dataset is slightly higher at 19.8% than that of metapath2vec at 19.1%, good noise control performance is achieved in all other datasets. For example, RFRL performance decreases by only 15.2% on Yelp. **(3)*In summary***, RFRL is more robust and more compatible with noise.

5.5.3 Scalability

In the era of big data, the scalability of the model is very important, so we further validate the scalability of the RFRL model. In this experiment, intra-type feature learning process is parallelized using multi-threading techniques, and the ratio of parallelized threads to accelerated multipliers is recorded. The experimental results are shown in Figure 9.

Figure 9 plots the number of threads *versus* the speedup multiplier. In the figure, the vertical axis is the acceleration multiplier, the horizontal axis is the number of concurrent threads, and the curve is the ratio between the two. By analyzing Figure 9, we can see that the RFRL model has a significant speedup with fewer threads. Specifically, a speedup of 10–12 times is achieved when using 16 threads for concurrent execution. As the number of threads increases, the speedup increases more slowly. For example, when 40 threads are executed concurrently, the speedup is only 18 to 19 times. In addition, the speedup almost stops growing when the experiment is performed concurrently with 78 threads.

5.6 Parameter sensitivity

The context window length L is an important hyper-parameter for our RFRL model, which determines the range of node intra-type proximity features needs to learn. In the type-aware biased sampling strategy, any two nodes in a context window constitute a positive sample. That is to say, if the window length L is larger, then the two nodes are farther away in the original network, and the order of proximity between them is higher. On the contrary, if the window length L is smaller, the closer the two nodes are in the original network, the lower the order of proximity between them is. In this experiment, the performance of our RFRL model with different window lengths is evaluated on the link prediction task using the ACC metric.

Figure 10 shows the ACC accuracy of our RFRL model with different window lengths L. From the figure, we can get the following observations. (1) *Focusing on a network*, the link prediction accuracy rises and then falls as the L value increases. For example, for the DBLP dataset, the ACC value increases rapidly in the range [1,2]. Then, in the range [2,4], the ACC values stabilize. Finally, in the range [4,7], the ACC value decreases slowly. This indicates that too small L values cannot fully capture the intra-type features of nodes in the network, and too large L leads to capturing imprecise intra-type features. Specifically, the relative stability range of the parameter L is [1,3] on Amazon, [2,4] on DBLP and Yelp. (2) *Comparing different datasets*, the window length of the Yelp and DBLP datasets should be larger than that of the Amazon dataset. This is because neighboring nodes in a walk sequence are more closely related in a dense network. It experimentally demonstrates that the first- and

second-order neighbor relations of nodes are the most worth learning, which can help the model to capture the proximity feature information between nodes well. (3) *In summary*, with context window length L in the range of [2,3], the model has the most stable performance for different datasets. In all our experiments, the value of the context window length is 2 as default.

6 Conclusion

In this paper, we propose a robust and fast representation learning model for heterogeneous networks, called RFRL. The RFRL model is well adapted to the following characteristics of future heterogeneous networks: larger scale, more diverse features, and stronger noise. *To better cope with large-scale networks*, two novel shallow learning strategies are designed to replace the traditional deep learning network to quickly generate the low-dimensional feature vectors of nodes. *To better learn complex and diverse features*, each node type and link type is treated as a feature subspace to perform representation learning separately. The RFRL model uses multiple learning processes for partial features instead of a single learning process for all features to achieve high speed and performance. *To reduce the impact of network noise*, GANs are further used to generate fake training samples in each subspace, and the adversarial learning between generator and discriminator can help the RFRL model to capture more robust and generalized node features. Extensive experimental results on multiple networks and multiple tasks demonstrate the performance of our model.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

Author contributions

LC: Conceptualization, Writing- Original draft preparation, Formal analysis, Supervision. YoL: Conceptualization, Validation, Writing- Reviewing and Editing, Investigation. YuL: Conceptualization, Validation, Visualization, Investigation. RX: Conceptualization, Supervision. Resources, Visualization. ZL: Methodology, Data curation, Validation. All authors have read and agreed to the published version of the manuscript.

Funding

This research was funded by the National Key Research and Development Program (No. 2019YFE0105300); the National Natural Science Foundation of China (No. 62103143), the Hunan Province Key Research and Development Program (No. 2022WK2006), the Special Project for the Construction of Innovative Provinces in Hunan (Nos. 2020TP2018 and 2019GK4030), the Young Backbone Teacher of Hunan Province (No. 2022101), and the Scientific Research Fund of Hunan Provincial Education Department (Nos. 22B0471 and 22C0829).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

1. He D, Liang C, Huo C, Feng Z, Jin D, Yang L, et al. Analyzing heterogeneous networks with missing attributes by unsupervised contrastive learning. *IEEE Trans Neural Networks Learn Syst* (2022) 2022:1–13. doi:10.1109/tnnls.2022.3149997

2. Huang F, Yi P, Wang J, Li M, Peng J, Xiong X. A dynamical spatial-temporal graph neural network for traffic demand prediction. *Inf Sci* (2022) 594:286–304. doi:10.1016/j.ins.2022.02.031

3. Amara A, Taieb MAH, Aouicha MB. Cross-network representation learning for anchor users on multiplex heterogeneous social network. *Appl Soft Comput* (2022) 118:108461. doi:10.1016/j.asoc.2022.108461

4. Bing R, Yuan G, Zhu M, Meng F, Ma H, Qiao S. Heterogeneous graph neural networks analysis: A survey of techniques, evaluations and applications. *Artif Intelligence Rev* (2022) 2022:1–40. doi:10.1007/s10462-022-10375-2

5. Pham P, Nguyen LT, Nguyen NT, Kozma R, Vo B. A hierarchical fused fuzzy deep neural network with heterogeneous network embedding for recommendation. *Inf Sci* (2023) 620:105–24. doi:10.1016/j.ins.2022.11.085

6. Fu Y, Yu X, Wu Y, Ding X, Zhao S. Robust representation learning for heterogeneous attributed networks. *Inf Sci* (2023) 628:22–49. doi:10.1016/j.ins.2023.01.038

7. Han K, Wang Y, Xu C, Guo J, Xu C, Wu E, et al. GhostNets on heterogeneous devices via cheap operations. *Int J Comp Vis* (2022) 130(4):1050–69. doi:10.1007/s11263-022-01575-y

8. Chen C, Li K, Wei W, Zhou JT, Zeng Z. Hierarchical graph neural networks for few-shot learning. *IEEE Trans Circuits Syst Video Tech* (2021) 32(1):240–52. doi:10.1109/tcsvt.2021.3058098

9. Chen L, Li Y, Deng X, Liu Z, Lv M, He T. Semantic-aware network embedding via optimized random walk and paragaraph2vec. *J Comput Sci* (2022) 63:101825. doi:10.1016/j.jocs.2022.101825

10. Dong Y, Chawla NV, Swami A. metapath2vec: Scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining; August 13-17, 2017; Halifax, NS, Canada (2017).

11. Shi Y, Gui H, Zhu Q, Kaplan L, Han J. Aspem: Embedding learning by aspects in heterogeneous information networks. In: Proceedings of the 2018 SIAM International Conference on Data Mining; May 3-5, 2018; San Diego (2018).

12. Bejani MM, Ghatee M. A systematic review on overfitting control in shallow and deep neural networks. *Artif Intelligence Rev* (2021) 54:6391–438. doi:10.1007/s10462-021-09975-1

13. Liu K, Zheng M, Liu Y, Yang J, Yao Y. Deep autoencoder thermography for defect detection of carbon fiber composites. *IEEE Trans Ind Inform* (2022) 2022:1. doi:10.1109/tii.2022.3172902

14. Zhang C, Song D, Huang C, Swami A, Chawla NV. Heterogeneous graph neural network. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining; August 4-8, 2019; Anchorage, AK, USA (2019).

15. Deng Z, Zhu Q, He P, Zhang D, Luo Y. A saliency detection and gram matrix transform-based convolutional neural network for image emotion classification. *Security Commun Networks* (2021) 2021:1–12. doi:10.1155/2021/6854586

16. Yu F, Liu L, Xiao L, Li K, Cai S. A robust and fixed-time zeroing neural dynamics for computing time-variant nonlinear equation using a novel nonlinear activation function. *Neurocomputing* (2019) 350:108–16. doi:10.1016/j.neucom.2019.03.053

17. Yu F, Shen H, Yu Q, Kong X, Sharma PK, Cai S. Privacy protection of medical data based on multi-scroll memristive hopfield neural network. *IEEE Trans Netw Sci Eng* (2022) 10:845–58. doi:10.1109/tnse.2022.3223930

18. Wang J, Wu Y, He S, Sharma PK, Yu X, Alfarraj O, et al. Lightweight single image super-resolution convolution neural network in portable device. *KSII Trans Internet Inf Syst* (2021) 15(11).

19. Chen L, Li Y, Deng X, Liu Z, Lv M, Zhang H. Dual auto-encoder GAN-based anomaly detection for industrial control system. *Appl Sci* (2022) 12(10):4986. doi:10.3390/app12104986

20. He Y, Song Y, Li J, Ji C, Peng J, Peng H. Hetespaceywalk: A heterogeneous spacey random walk for heterogeneous information network embedding. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management; November 3 - 7, 2019; Beijing China (2019).

21. Wang X, Zhang Y, Shi C. Hyperbolic heterogeneous information network embedding. In: Proceedings of the AAAI conference on artificial intelligence; 27 January 2019- 1 February 2019; Hawaii USA (2019).

22. Jiang J-Y, Li Z, Ju CJ-T, Wang W. Maru: Meta-context aware random walks for heterogeneous network representation learning. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management; 19th to the 23rd of October 2020 (2020).

23. Xu L, Wei X, Cao J, Yu PS. Embedding of embedding (EOE) joint embedding for coupled heterogeneous networks. In: Proceedings of the tenth ACM international conference on web search and data mining; Feb 06, 2017-Feb 10, 2017; Cambridge, United Kingdom (2017).

24. Lu Y, Shi C, Hu L, Liu Z. Relation structure-aware heterogeneous information network embedding. In: Proceedings of the AAAI Conference on Artificial Intelligence; February 22-March 1, 2022 (2019).

25. Li B, Pi D, Lin Y, Khan IA, Cui L. Multi-source information fusion based heterogeneous network embedding. *Inf Sci* (2020) 534:53–71. doi:10.1016/j.ins.2020.05.012

26. Chen H, Yin H, Wang W, Wang H, Nguyen QVH, Li X. Pme: Projected metric embedding on heterogeneous networks for link prediction. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining; August 19 - 23, 2018; London United Kingdom (2018).

27. Wang J, Zou Y, Lei P, Sherratt RS, Wang L, Hu F, et al. Changes in colistin resistance and mcr-1 abundance in *Escherichia coli* of animal and human origins following the ban of colistin-positive additives in China: An epidemiological comparative study. *J Internet Tech* (2020) 21(4):1161–71. doi:10.1016/S1473-3099(20)30149-3

28. Fu X, Zhang J, Meng Z, King I. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In: Proceedings of The Web Conference; April 20 - 24, 2020; Taipei Taiwan (2020).

29. Wang X, Ji H, Shi C, Wang B, Ye Y, Cui P, et al. Heterogeneous graph attention network. In: The world wide web conference; May 13 to May 17, 2019; San Francisco, CA, USA (2019).

30. Long M, Zeng Y. Detecting iris liveness with batch normalized convolutional neural network. *Comput Mater Contin* (2019) 58(2):493–504. doi:10.32604/cmc.2019.04378

31. Zhao K, Bai T, Wu B, Wang B, Zhang Y, Yang Y, et al. Deep adversarial completion for sparse heterogeneous information network embedding. In: Proceedings of The Web Conference; April 20 - 24, 2020; Taipei Taiwan (2020).

32. Hu Z, Dong Y, Wang K, Sun Y. Heterogeneous graph transformer. In: Proceedings of the web conference; April 20 - 24, 2020; Taipei Taiwan (2020).

33. Hu B, Fang Y, Shi C. Adversarial learning on heterogeneous information networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; August 4-8, 2019; Anchorage, AK, USA (2019).

34. Hong H, Guo H, Lin Y, Yang X, Li Z, Ye J. An attention-based graph neural network for heterogeneous structural learning. In: Proceedings of the AAAI conference on artificial intelligence (2020).