

# NoiseHGNN: Synthesized Similarity Graph-Based Neural Network For Noised Heterogeneous Graph Representation Learning

Xiong Zhang<sup>1</sup>, Cheng Xie<sup>1\*</sup>, Haoran Duan<sup>2</sup>, Beibei Yu<sup>3</sup>

<sup>1</sup> School of Software, Yunnan University, Kunming, China

<sup>2</sup> School of Cyber Science and Engineering, Wuhan University, Wuhan, China

<sup>3</sup> Australian AI Institute, University of Technology Sydney, Sydney, Australia

{zhangxiong@stu, xiecheng}@ynu.edu.cn, hrduan07@gmail.com, Beibei.Yu@student.uts.edu.au

## Abstract

Real-world graph data environments intrinsically exist noise (e.g., link and structure errors) that inevitably disturb the effectiveness of graph representation and downstream learning tasks. For homogeneous graphs, the latest works use original node features to synthesize a similarity graph that can correct the structure of the noised graph. This idea is based on the homogeneity assumption, which states that similar nodes in the homogeneous graph tend to have direct links in the original graph. However, similar nodes in heterogeneous graphs usually do not have direct links, which can not be used to correct the original noise graph. This causes a significant challenge in noised heterogeneous graph learning. To this end, this paper proposes a novel synthesized similarity-based graph neural network compatible with noised heterogeneous graph learning. First, we calculate the original feature similarities of all nodes to synthesize a similarity-based high-order graph. Second, we propose a similarity-aware encoder to embed original and synthesized graphs with shared parameters. Then, instead of graph-to-graph supervising, we synchronously supervise the original and synthesized graph embeddings to predict the same labels. Meanwhile, a target-based graph extracted from the synthesized graph contrasts the structure of the metapath-based graph extracted from the original graph to learn the mutual information. Extensive experiments in numerous real-world datasets show the proposed method achieves state-of-the-art records in the noised heterogeneous graph learning tasks. In highlights, +5~6% improvements are observed in several noised datasets compared with previous SOTA methods.

## 1 Introduction

Graph representation learning is one of the most significant research fields in artificial intelligence, since most intelligent applications are based on graph representations such as recommendation systems (Lv et al. 2021; Fan et al. 2019; Zhao et al. 2017), social networks (Qiu et al. 2018; Li and Goldwasser 2019; Wang et al. 2019a), biomedicine (Gaudelet et al. 2021; Fout et al. 2017; Davis et al. 2019), fraud detection (Shchur et al. 2018; Dou et al. 2020), e-commerce (Ji et al. 2021; Zhao et al. 2019), etc. However, real-world graph data environments inherently contain noise data (e.g.,

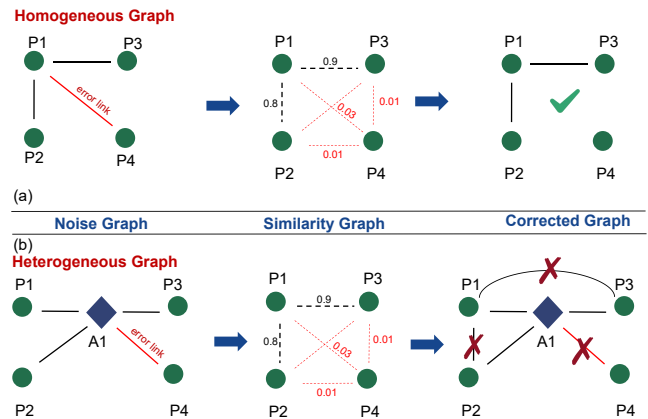


Figure 1: Homogeneity assumption works for noised homogeneous graph but failed in noised heterogeneous graph.

structure errors, etc.) that challenge the representation models. Specifically, representation models are required to maintain their effectiveness in noised graph data environments.

The latest works attempting to solve this problem by optimizing graph structure (Wang et al. 2021; Wei et al. 2022) or synthesizing graph structure (Fatemi, El Asri, and Kazemi 2021; Franceschi et al. 2019; Jin et al. 2020; Liu et al. 2022) to alleviate the noise. Nowadays, graph synthesizing-based methods achieve impressive performance in noised homogeneous graph representation learning, based on the homogeneity assumption to create a synthesized graph to correct the noise homogeneous graph. As an example shown in Fig.1 (a), a paper citation graph has four paper nodes: P1, P2, P3, and P4. The links between nodes represent the references among the papers. The red link means P1 and P4 are incorrectly realized to have a reference relationship. To fix these error links, a similarity graph can be constructed in advance by calculating the node-to-node feature similarities without using links. It can be observed that P1 and P4 have minimal similarity values (0.03), which means P1 is unlikely to cite P4 in the original graph. Thus, the original noised graph will likely be corrected by calculating the loss between it and the similarity graph. This idea is based on the homogeneity assumption, which states that similar nodes tend to have direct links in a homogeneous graph.

\*Corresponding author.

However, the above solution can not be simply applied to the heterogeneous graphs. As an example shown in Fig.1 (b), a heterogeneous graph represents the relationships between papers and authors. A similarity graph shows that P1, P2, and P3 are similar enough to have a link. We will obtain a wrong correction result if we directly calculate the loss between the original and similarity graphs. This is because links in a heterogeneous graph do not represent “similar” semantics. Different types of links represent different semantics that need to be specifically dealt with. Current metapath-dependent (Wang et al. 2019b; Fu et al. 2020) or edge-relationship-oriented (Zhou et al. 2023; Lv et al. 2021; Zhao et al. 2022; Zhu et al. 2019; Zhang et al. 2019; Schlichtkrull et al. 2018; Du et al. 2023) models prone to misconnections in links when there are errors in the graph structure, leading to effectiveness degradation. Thus, maintaining representation models’ effectiveness in noised heterogeneous graphs is still challenging.

To address this challenge, we propose a novel Noise-HGNN model compatible with noised heterogeneous graph learning. Since the similarity and heterogeneous graphs represent different semantics, we do not calculate their mutual information to correct the noised graph. Instead, two novel modules, Similarity-aware HGNN, and Metapath-Target contrastive learning, are proposed. In the Similarity-aware HGNN module, the synthesized similarity graph is used to reinforce the node attention during the representation learning instead of directly supervising the noised graph. In the Metapath-Target contrastive module, instead of directly contrasting synthesized and noised graphs, we contrast the metapath-based graph (extracted from the noised graph) and the target-based graph (extracted from the synthesized graph) that represent the same semantics. Then, the representations of both the noised and synthesized graphs are jointly utilized to predict the labels during model training. Finally, the noised graph representation is used to predict the label during the testing. Extensive experiments on five real-world datasets demonstrate the effectiveness and generalization ability of NoiseHGNN. The proposed Noise-HGNN achieves state-of-the-art records on four out of five extensively benchmark datasets under the noised data environment. In highlights, in the complex and noise-sensitive dataset (DBLP, PubMed, and IMDB), +3 ~ +6% improvement is observed compared with peer methods. In the rest of the datasets, +1 ~ +2% improvement is observed compared with peer methods. The code and datasets are available at <https://github.com/kg-cc/NoiseHGNN>.

In summary, our contributions are as follows:

- It is the first work, to our best knowledge, to investigate noised heterogeneous graph representation learning and achieves state-of-the-art records.
- The proposed Metapath-Target contrastive learning bridges the homogeneity assumption to the noised heterogeneous graph representation.
- The proposed similarity-aware HGNN takes advantage of the similarity graph from traditional homogeneous graphs to the noised heterogeneous graph representation learning.

## 2 Related Work

### 2.1 Graph Structure Learning

For homogeneous graphs, existing methods address erroneous link perturbations by constructing and refining the graph structure based on the homogeneous graph assumption through graph structure learning. These methods correct the structure of noisy graphs by synthesizing similarity graphs using features of the original nodes. Specifically, existing methods employ probabilistic models (Wang et al. 2021; Franceschi et al. 2019) and metric learning models (Fatemi, El Asri, and Kazemi 2021; Franceschi et al. 2019; Jin et al. 2020; Liu et al. 2022) to parameterize the adjacency matrix and jointly optimize the parameters of the adjacency matrix and GNNs by solving downstream models. However, in heterogeneous graphs, similar nodes are often not directly connected, making it difficult to use these methods to correct the original noisy graph. For heterogeneous graphs, HGSL (Zhao et al. 2021) simultaneously performs Heterogeneous Graph Structure Learning and GNN parameter learning for classification. However, it does not account for the potential noise introduced by erroneous edges within the heterogeneous graph structure.

### 2.2 Heterogeneous Graph Neural Network

HGNNs can generally be categorized into two types based on their strategies for handling heterogeneity: message-passing based HGNNs, meta-paths based HGNNs.

**Message-passing based HGNNs.** RGCN (Schlichtkrull et al. 2018) assigns different weight matrices to various relation types and aggregates one-hop neighbors. RSHN (Zhu et al. 2019) builds a coarsened line graph to get edge features and adopts message passing to propagate node and edge features. SimpleHGN (Lv et al. 2021) incorporates relational weight matrices and embeddings to characterize heterogeneous attention at each edge. Additionally, Space4HGNN (Zhao et al. 2022) defines a unified design space for HGNNs to exhaustively evaluate combinations of multiple technologies. SlotGAT (Zhou et al. 2023) designs a slot for each type of node according to the node type and uses the slot attention mechanism to construct the HGNN model.

**Meta-paths based HGNNs.** Another class of HGNNs captures higher-order semantic information through meta-paths. HAN (Wang et al. 2019b) employs hierarchical attention mechanisms to capture both node-level importance between nodes and the semantic-level importance of meta-paths. MAGNN (Fu et al. 2020) enhances this approach with several meta-path encoders to comprehensively encode information along meta-paths. In contrast, the Graph Transformation Network (Yun et al. 2019) (GTN) can automatically learn meta-paths through graph transformation layers. However, for heterogeneous graphs with multiple edge types, meta-path-based methods are less practical due to the high cost of acquiring meta-paths. HetGNN (Zhang et al. 2019) addresses this issue by using random walks to sample fixed-size neighbors for nodes of different types and then applying recurrent neural networks (RNNs) for representation learning. Seq-HGNN (Du et al. 2023) designs a sequential node representation learning mechanism to represent each node

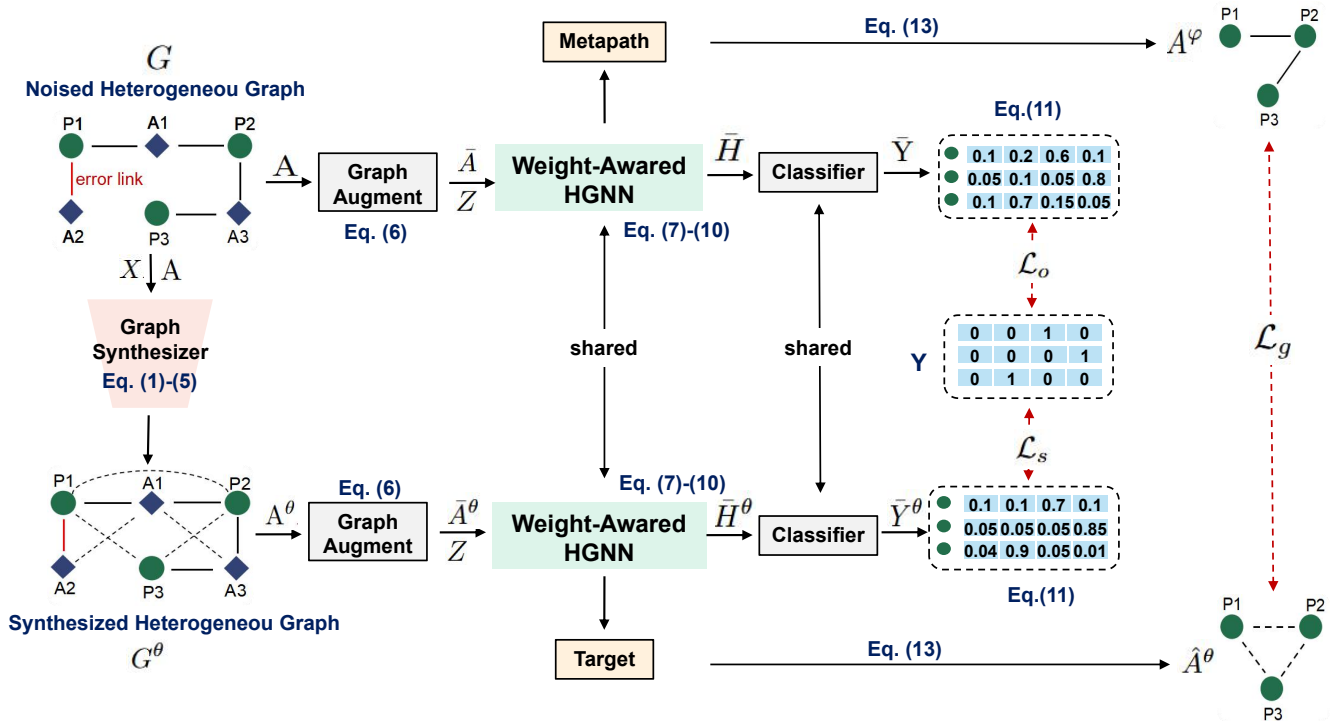


Figure 2: The overall framework of the proposed model.

as a sequence of meta-path representations during the node message passing.

### 3 Preliminaries

#### 3.1 Heterogeneous Graph

A heterogeneous graph (Sun and Han 2012) can be defined as  $G = \{\mathcal{V}, \mathcal{E}, \phi, \psi\}$ , where  $\mathcal{V}$  is the set of nodes and  $\mathcal{E}$  is the set of edges. Each node  $v$  has a type  $\phi(v)$  and each edge  $e$  has a type  $\psi(e)$ . The sets of possible node types and edge types are denoted by  $T_v = \{\phi(v) : \forall v \in \mathcal{V}\}$  and  $T_e = \{\psi(e) : \forall e \in \mathcal{E}\}$ , respectively. For a heterogeneous graph  $|\psi| + |\phi| > 2$ . When  $|T_v| = |T_e| = 1$ , the graph degenerates into an ordinary homogeneous graph.

A node  $v$  has a feature vector  $\mathbf{x}_v$ . For node type  $t \in \Phi$ , all type- $t$  nodes  $v \in \{v \in \mathcal{V} | \phi(v) = t\}$  have the same feature dimension  $d_0^t = d_0^{\phi(v)}$ , i.e.,  $\mathbf{x}_v \in \mathbb{R}^{d_0^{\phi(v)}}$ . Nodes of different types can have different feature dimensions (Lv et al. 2021). For input nodes feature type, we use  $\eta = 0$  to denote using all given features,  $\eta = 1$  to denote using only target node features, and  $\eta = 2$  to denote all nodes with one-hot features.

#### 3.2 Noised Heterogeneous Graph

Let  $G = \{\mathcal{V}, \mathcal{E}, \phi, \psi\}$  denote a heterogeneous graph.  $N$  be the number of nodes, i.e.,  $N = |\mathcal{V}|$ , and  $M$  be the number of links, i.e.,  $M = |\mathcal{E}|$ . The sets of node types is denoted by  $T_v = \{\phi(v) : \forall v \in \mathcal{V}\}$ . An edge  $e$  corresponds to an edge type  $\psi(e)$  connecting two types of nodes  $v_i$  and  $v_j$ , with the node types  $\phi(v_i)$  and  $\phi(v_j)$  respectively.

We simulate the erroneous link scenario in real data by modifying the target node  $v_j$  from  $T_{v_j}$  connected to  $v_i$ . Specifically, for all datasets, we randomly modify the target nodes of 30% of the  $M$  links, preserving the heterogeneous graph edge type  $\psi(e)$ . This ensures that the modified edge types remain consistent with their original types.

### 4 The Proposed Method

The proposed model mainly consists of four modules: a graph synthesizer, a graph augmter, a similarity-aware graph encoder, and a graph contrastive module, as shown in Fig 2. The synthesized similarity graph contains high-order semantics used to adapt the noise link weights by the similarity-aware graph encoder. The contrastive learning between the target graph (from the synthesized graph) and the meta-path graph (from the noised graph) can further alleviate noise links.

#### 4.1 Similarity Graph Synthesizer

In this work, the synthesized graph is a similarity graph. The base assumption of the similarity graph is that if two nodes are similar, they will likely have natural relationships. Thus, the similarity graph can potentially supervise the noise graph. Two processes are essential to synthesize a similarity graph: (1) node feature projection and (2) similarity adjacent matrix synthesizing.

**Node Feature Projection.** The node features need to be projected into a unified feature space due to the heterogeneity of nodes. Therefore, we design type-specific linear transformations to project the features of different types of nodes

$X^\phi$  into the unified feature space  $Z$ , as defined in equation 1.

$$Z = \begin{bmatrix} Z^{\phi_1} = W^{\phi_1} X^{\phi_1} + b^{\phi_1} \\ Z^{\phi_2} = W^{\phi_2} X^{\phi_2} + b^{\phi_2} \\ \vdots \\ Z^{\phi_m} = W^{\phi_m} X^{\phi_m} + b^{\phi_m} \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{bmatrix} \quad (1)$$

where  $\phi_m$  denotes the node type,  $X^{\phi_m}$  represents the feature matrix of the node type  $\phi_m$ ,  $W^{\phi_m}$  is a learnable matrix,  $b^{\phi_m}$  denotes vector bias. We transform the node features using type-specific linear transformations for nodes with features and use  $X^{\phi_m} \in \mathbb{R}^{N \times d_0^{\phi_m}}$  to denote the node feature matrix and  $Z \in \mathbb{R}^{N \times d'}$  denotes the transformed node features.

**Graph Feature Projection.** The graph projection projects the graph-independent feature  $Z$  into the graph-dependent feature  $\bar{Z}$ . Let  $\eta$  as an indicator to categorize the graph data environment on feature types.  $\eta = 0$  indicates that all attributes of nodes are embedded into the node features (e.g., title, description, abstract, and name of a node).  $\eta = 1$  means only the target node’s attributes are embedded into the node features.  $\eta = 2$  denotes the node’s one-hot attributes are used. According to the value of  $\eta$ , the graph feature projection is defined as equation 2.

$$\bar{Z} = \begin{cases} \sigma(ZW + b), & \eta \in [0] \\ \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} ZW), & \eta \in [1, 2] \end{cases}, \bar{Z} = \begin{bmatrix} \bar{z}_1 \\ \vdots \\ \bar{z}_N \end{bmatrix} \quad (2)$$

where  $W$  is the parameter matrix,  $\sigma$  is a non-linear function that makes training more stable,  $\tilde{A} = A + I$  is the adjacency matrix with self-loop while  $\tilde{D}$  is the degree matrix of  $\tilde{A}$ .

When  $\eta = 0$ , a simple linear projection is applied. This is because, in this case, the node features already contain the first-order graph features (i.e., attributes of the node). When  $\eta = 1, 2$ , a typical graph convolutional network is used to aggregate topology information into the node feature. This is because, in this case, the node itself contains no semantics that need to be enriched from the graph structure.

**Similarity Adjacent Matrix Synthesizing.** To construct the adjacent matrix, we first calculate the similarity value node-to-node based on the graph-dependent node feature  $\bar{Z}$ . The equation 3 presents the process for calculating the node similarity.

$$S_{i,j} = \frac{\bar{z}_i \cdot \bar{z}_j}{|\bar{z}_i| \cdot |\bar{z}_j|} \quad (3)$$

where  $S_{i,j}$  is the cosine similarity between feature  $\bar{z}_i$  and  $\bar{z}_j$ .  $S \in \mathbb{R}^{N \times N}$ , and  $N$  is the total number of graph node.

Similarity adjacent matrix  $S$  is usually dense and represents fully connected graph structures, which are often not meaningful for most applications and can lead to expensive computational costs (Wang et al. 2021). Therefore, we apply the k-nearest neighbors (kNN)-based sparsification on

$S$ . Specifically, we retain the links with the top-k connection values for each node and set the rest to zero. Let  $A^\theta$  represent the Similarity adjacent matrix, as defined in equation 4.

$$A_{ij}^\theta = \begin{cases} S_{ij}, & S_{ij} \in \text{top-k}(S_i), \\ 0, & S_{ij} \notin \text{top-k}(S_i), \end{cases} \quad (4)$$

where  $\text{top-k}(S_i)$  is the set of top-k values of row vector  $S_i$ . For large-scale graphs, we perform the kNN sparsification with its locality-sensitive approximation (Fatemi, El Asri, and Kazemi 2021) where the nearest neighbors are selected from a batch of nodes instead of all nodes, reducing the memory requirement.

At last, combining the graph-independent node feature  $Z$  and the synthesized adjacent matrix  $A^\theta$ , the synthesized similarity graph  $G^\theta$  can be expressed as follows:

$$G^\theta = \{Z, A^\theta\} \quad (5)$$

## 4.2 Graph Augmentation

Augmentation is widely used in graph contrastive learning and representation learning. It can enhance mutual information and improve the model’s generalization ability. In this work, we apply the masking mechanism to augment the graph. In detail, for a given adjacency matrix  $A$ , we first sample a masking matrix  $M \in \{0, 1\}^{N \times N}$ , where each element of  $M$  is drawn independently from a Bernoulli distribution with probability  $p^{(A)}$ . In NoiseHGNN, we use this graph enhancement scheme to generate enhanced graphs from both the noise and synthesized graphs. The adjacency matrix is then masked with  $M$  and  $M^\theta$ :

$$\begin{aligned} \bar{A} &= A \odot M \\ \bar{A}^\theta &= A^\theta \odot M^\theta \end{aligned} \quad (6)$$

where  $\bar{A}$  and  $\bar{A}^\theta$  are the augmented noise graph and augmented synthesized graph, respectively. To obtain different context structures in the two views, edge discarding for the two views is performed with different probabilities  $p^{(A)} \neq p^{(A^\theta)}$ . Other advanced enhancement schemes can also be applied to NoiseHGNN, which is left for future research.

## 4.3 Similarity-Aware HGNN Encoder

The base idea of the similarity-aware encoder is to aggregate the neighbor feature through attention and similarity. In detail, the primary process of the similarity-aware encoder is (1) correlation coefficient, (2) similarity-aware attention, and (3) attention-based aggregation.

**Correlation coefficient.** The same as the classic Graph Attention Network (GAT), we first calculate the correlation coefficient  $e_{i,j}$  between a node  $i$  and its neighbors  $j$ , as the equation 7 shows.

$$\begin{aligned} e_{i,j} &= a([Wz_i || Wz_j]), j \in \text{Index}(\bar{A}_i) \\ e_{i,j}^\theta &= a([Wz_i || Wz_j]), j \in \text{Index}(\bar{A}_i^\theta) \end{aligned} \quad (7)$$

Here,  $W$  is a learnable parameter, while  $z$  is the projected node feature.  $a(\cdot)$  is a projection network that projects concatenated features to a scalar number ranging from 0 to 1.0.  $\text{Index}(\bar{A}_i)$  denotes a set that contains all indexes of nodes that are neighbors of node  $i$  in the noise graph.

$\text{Index}(\bar{A}_i^\theta)$  denotes a set that contains all indexes of nodes that are neighbors of node  $i$  in the synthesized similarity graph  $G^\theta$ .

**Similarity-aware attention.** Similar to GAT, we calculate the attention  $\alpha_{i,j}$  through the  $\text{Softmax}(\cdot)$  operation on correlation coefficient  $e_{i,j}$ . However, the difference is that the proposed attention also considers the similarity graph's adjacent matrix, as shown in equation 8.

$$\begin{aligned}\alpha_{i,j} &= \frac{\exp(\text{LeakyReLU}(e_{i,j}))}{\sum_{k \in \text{Index}(\bar{A}_i)} \exp(\text{LeakyReLU}(e_{i,k}))} \cdot \bar{A}_{i,j} \\ \alpha_{i,j}^\theta &= \frac{\exp(\text{LeakyReLU}(e_{i,j}^\theta))}{\sum_{k \in \text{Index}(\bar{A}_i^\theta)} \exp(\text{LeakyReLU}(e_{i,k}^\theta))} \cdot \bar{A}_{i,j}^\theta\end{aligned}\quad (8)$$

From the above attention, the similar node pairs will have a higher attention coefficient, while the dissimilar node pairs will only have zero attention value. This potentially intercepts the message propagation on error-prone links when the corresponding attention is zero.

**Attention-based aggregation.** The aggregation aims to obtain the graph representation  $H$  from node feature  $Z$  and adjacent matrix  $A$ . Let  $W^\alpha$  denote a learnable parameter of the aggregation process. The noise and synthesized similarity graphs share the same  $W^\alpha$ .  $\sigma(\cdot)$  denote a  $\text{LeakyReLU}(\cdot)$  operation. Equation 9 defines the node representation calculated by similarity-aware attention  $\alpha$  and node feature  $Z$ .

$$\begin{aligned}\bar{h}_i &= \sigma\left(\sum_{j \in \text{Index}(\bar{A}_i)} \alpha_{ij} \cdot W^\alpha \cdot z_j\right) \\ \bar{h}_i^\theta &= \sigma\left(\sum_{j \in \text{Index}(\bar{A}_i^\theta)} \alpha_{ij}^\theta \cdot W^\alpha \cdot z_j\right)\end{aligned}\quad (9)$$

where  $j \in \text{Index}(\bar{A}_i)$  and  $j \in \text{Index}(\bar{A}_i^\theta)$  represent the neighbors of node  $i$  under the noise and synthesized graphs respectively.  $\bar{h}_i$  and  $\bar{h}_i^\theta$  represent the node representations of node  $v_i$  under the noise and synthesized similarity graphs, respectively.

Finally, all nodes get the graph representation  $H$ , which can be represented as follows:

$$\bar{H} = \begin{bmatrix} h_1 \\ \vdots \\ h_N \end{bmatrix}, \bar{H}^\theta = \begin{bmatrix} h_1^\theta \\ \vdots \\ h_N^\theta \end{bmatrix}\quad (10)$$

where  $\bar{H}$  denotes all node representations under the noise graph, while  $\bar{H}^\theta$  denotes all node representations under the synthesized similarity graph.

#### 4.4 Graph Learning

**Learning Objective.** So far, we have the graph representations  $\bar{H}$  and  $\bar{H}^\theta$ . The representations can be used for multiple downstream objectives such as node classifications,

graph classifications, clustering, link prediction, etc. This work focuses on node classifications, which categorize a node instance into a class label. Let  $\text{MLP}(\cdot)$  represent a linear neural network. The learning objective (node classification) is defined as follows:

$$\begin{aligned}\bar{Y} &= \text{Softmax}(\text{MLP}(\bar{H})), \\ \bar{Y}^\theta &= \text{Softmax}(\text{MLP}(\bar{H}^\theta))\end{aligned}\quad (11)$$

where  $\bar{Y}$  and  $\bar{Y}^\theta$  are the predictions obtained under the noise graph and the synthesized similarity graph, respectively.

**Classification loss.** The datasets normally have two characteristics: single-label and multi-label. For single-label classification, softmax and cross-entropy loss are used. For multi-label datasets, sigmoid activation and binary cross-entropy loss are used. Let  $\mathcal{L}_{CE}(\cdot)$  represent the cross-entropy for both single-label and multi-label in general.  $Y$  is the ground truth label of the classification. The classification loss is defined as equation 12.

$$\begin{aligned}\mathcal{L}_o &= \mathcal{L}_{CE}(\bar{Y}, Y) \\ \mathcal{L}_s &= \mathcal{L}_{CE}(\bar{Y}^\theta, Y)\end{aligned}\quad (12)$$

where  $\mathcal{L}_o$  is the classification loss for the noise graph representation.  $\mathcal{L}_s$  is the classification loss for the synthesized similarity graph representation.

**Contrastive Loss.** To better mitigate the error links in the noise graph structure, we extract the meta-path  $A^\varphi$  graph from the noise graph structure (e.g., author-paper-author (APA) in the DBLP) and also extract the target graph  $\hat{A}^\theta$  from the synthesized similarity graph, as presented in equation 13.

$$\begin{aligned}A^\varphi &= \text{MetaPath}(A) \\ \hat{A}^\theta &= \text{Target}(A^\theta)\end{aligned}\quad (13)$$

Based on the above subgraphs, we then use the scaled cosine loss function to calculate graph contrastive loss  $\mathcal{L}_g$ . Since the higher-order synthesized similarity graph is an adjacency matrix with higher-order semantic information, it can alleviate the errors in the meta-paths graph. Specifically, the contrastive loss  $\mathcal{L}_g$  is calculated as follows:

$$\mathcal{L}_g = \left(1 - \frac{(A^\varphi)^T \cdot \hat{A}^\theta}{\|(A^\varphi)^T\| \cdot \|\hat{A}^\theta\|}\right)^\gamma, \gamma \geq 1\quad (14)$$

where the scaling factor  $\gamma$  is a hyper-parameter adjustable over different datasets,  $A^\varphi$  denotes the graph obtained from the noise graph via meta-paths and  $\hat{A}^\theta$  denotes the target graph extracted from the synthesized graph.

**Final Loss.** Combining equation 12 and 13, the final loss function is formulated as:

$$\mathcal{L} = \mathcal{L}_o + \mathcal{L}_s + \mathcal{L}_g\quad (15)$$

The whole process of the above equations is organized as a pseudo-code algorithm, see the appendix for details.

	DBLP(0.3)		IMDB(0.3)		ACM(0.3)		PubMed_NC(0.3)		Freebase(0.3)	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
GCN (2016)	50.97±0.14	52.35±0.17	37.86±3.93	50.60±1.73	86.38±1.21	86.46±1.14	36.28±2.78	41.39±2.81	17.16±0.22	53.66±0.18
GAT (2017)	63.11±1.48	64.38±1.23	38.53±5.65	49.65±4.58	79.02±3.35	79.61.60±3.26	34.08±4.66	39.53±3.89	17.83±0.73	54.63±0.23
RGCN (2018)	43.02±1.83	45.02±1.53	39.79±1.77	47.28±0.90	54.37±2.38	55.36±2.25	15.28±3.30	19.76±3.96	-	-
RSHN (2019)	67.01±1.47	67.69±1.39	31.40±3.92	48.09±1.45	80.43±2.09	80.43±1.83	-	-	-	-
HAN (2019)	62.32±1.79	63.43±1.59	38.64±4.7	49.93±2.97	78.31±1.29	78.19±1.37	-	-	-	-
HetGNN (2019)	59.31±0.23	60.23±0.36	37.36±0.75	40.85±0.86	74.45±0.13	74.39±0.15	-	-	-	-
MAGNN (2020)	60.83±1.40	62.66±1.07	14.30±0.20	39.98±0.12	88.67±0.84	88.64±0.73	-	-	-	-
HGT (2020)	48.11±4.39	53.66±2.54	45.26±0.89	55.62±0.16	81.52±1.82	81.43±1.77	<u>46.65±3.63</u>	49.53±2.61	10.40±0.94	49.31±0.13
simpleHGN (2021)	65.54±1.03	67.63±0.75	<u>58.21±2.20</u>	<u>62.93±0.92</u>	89.65±0.52	89.64±0.48	42.25±4.67	47.90±3.40	27.10±1.54	<u>56.79±0.48</u>
space4HGNN(2022)	65.43±1.12	67.81±0.55	55.43±0.54	61.91±0.94	88.52±0.74	88.63±0.34	42.41±3.59	47.14±3.16	26.93±0.78	53.54±0.33
Seq-HGNN(2023)	67.89±0.26	68.93±0.37	56.23±0.43	62.34±0.29	90.02±0.23	90.09±0.31	44.41±4.31	48.54±4.01	27.93±0.84	55.94±0.48
SlotGAT(2024)	<u>68.23±0.75</u>	<u>69.77±0.19</u>	54.89±1.77	62.47±0.97	<b>90.64±0.46</b>	<b>90.65±0.42</b>	44.76±3.36	<u>50.88±1.54</u>	<u>28.14±1.31</u>	56.56±0.64
NoiseHGNN	<b>71.96±0.33</b>	<b>73.16±0.19</b>	<b>60.46±0.69</b>	<b>63.94±0.26</b>	<u>90.16±0.49</u>	<u>90.10±0.46</u>	<b>50.92±3.52</b>	<b>55.81±3.20</b>	<b>30.57±2.94</b>	<b>56.96±0.67</b>

Table 1: Peer comparison on node classification task. 30% error link rate is applied to the datasets. Vacant positions (“-”) mean out of memory in our computational environment. The best record is marked in bold, and the runner-up is underlined.

## 5 Experiments

### 5.1 Experiment Settings

**Datasets.** Table 2 reports the statistics of the benchmark datasets widely used in previous studies (Lv et al. 2021; Zhou et al. 2023; Zhao et al. 2022). These datasets span various domains, such as academic graphs (e.g., DBLP, ACM), information graphs (e.g., IMDB, Freebase), and medical-biological graphs (e.g., PubMed). For node classification, each dataset contains a target node type, and all nodes of this target type are used for classification.

Node Classification	Nodes	Node Types	Edges	Edge Types	error link	Target	Classes
DBLP	26,128	4	239,566	6	30%	author	4
IMDB	21,420	4	86,642	6	30%	movie	5
ACM	10,942	4	547,872	8	30%	paper	3
Freebase	180,098	8	1,057,688	36	30%	book	7
PubMed_NC	63,109	4	244,986	10	30%	disease	8

Table 2: Statistics of Datasets.

**Baselines.** We compare our method with several state-of-the-art models, including HAN (Wang et al. 2019b), MAGNN (Fu et al. 2020), HetGNN (Zhang et al. 2019), HGT (Hu et al. 2020), RGCN (Schlichtkrull et al. 2018), RSHN (Zhu et al. 2019), SimpleHGN (Lv et al. 2021), Space4HGNN (Zhao et al. 2022), Seq-HGNN (Du et al. 2023) and SlotGAT (Zhou et al. 2023). Additionally, we include comparisons with GCN (Kipf and Welling 2016) and GAT (Veličković et al. 2017).

**Evaluation Settings.** For node classification, following the methodology described by Lv et al. (2021), we split the labeled training set into training and validation subsets at 80%:20%. The testing data are fixed, with detailed numbers provided in the appendix. For each dataset, we conduct experiments on five random splits, reporting the average and standard deviation of the results. For our methods, we perform a grid search to select the best hyperparameters on the validation set. The results of the node classification are reported using the averaged Macro-F1 and Micro-F1 scores.

### 5.2 Main Results

Table 1 presents the node classification results on a 30% noise data environment compared with peer methods. The comparison metrics are mean Macro-F1 and Micro-F1 scores. It can be observed that NoiseHGNN consistently outperforms all baselines for Macro-F1 and Micro-F1 on DBLP, IMDB, Pubmed, and Freebase. Overall, NoiseHGNN achieves an average +4.2% improvement on Macro-F1 and +2.6% on Micro-F1 compared with runner-up. In highlight, NoiseHGNN achieves the best results on IMDB and Pubmed datasets, with an average improvement of +5.9%. This is because IMDB and Pubmed are noise-sensitive datasets since their node features are just one-hot embedding, which contains no semantics. All the information is stored in the topology of the graphs, which have 30% error links. In the medium-size dataset, DBLP, NoiseHGNN performs stable, with a +3.5% improvement compared with the previous SOTA method. In the large and non-trivial dataset, Freebase, NoiseHGNN still obtained 30.57% Macro-F1, which is 2.4% higher than the previous SOTA method. Notably, NoiseHGNN gets the runner-up results (90.16%) in the ACM dataset, with -0.48% lower than SlotGAT (90.64%). This is because ACM is a noise-insensitive and trivial dataset, while the latest methods can achieve 90% and even higher results. Its nodes contain rich semantic embedding that can be simply used for node classification without using graph links.

### 5.3 Ablation Study

In this section, we demonstrate the effectiveness of NoiseHGNN through ablation experiments with w/o Graph Synthesized and Meta-Target Graph. As shown in table3, NoiseHGNN with graph synthesized and Meta-target graph achieves superior results across all datasets. Specifically, on the smaller DBLP dataset, NoiseHGNN with the graph synthesizer and Meta-target graph improves Macro-F1 and Micro-F1 scores by 2.41% and 3.81%, respectively. On the largest dataset, Freebase, the improvements are 11.40% in Macro-F1 and 3.24% in Micro-F1. Notably, on the IMDB dataset, NoiseHGNN achieves significant improvements of 42.01% in Macro-F1 and 25.83% in Micro-F1. These results

demonstrate that the higher-order synthesized graphs generated by the graph synthesizer effectively mitigate the negative effects of erroneous links, leading to substantially better performance.

Datasets		w/o Graph Synthesizer	w/o Meta-Target Graph	w/ All
DBLP	Macro-F1	70.26±1.23	71.66±0.58	<b>71.96±0.33</b>
	Micro-F1	70.47±1.28	72.78±0.58	<b>73.16±0.19</b>
IMDB	Macro-F1	43.46±6.58	59.77±1.77	<b>60.46±0.69</b>
	Micro-F1	50.78±1.29	63.46±0.73	<b>63.90±0.26</b>
ACM	Macro-F1	89.40±0.92	89.32±0.61	<b>90.16±0.49</b>
	Micro-F1	89.34±1.09	89.42±0.60	<b>90.10±0.46</b>
Pubmed_NC	Macro-F1	49.92±1.32	49.40±1.74	<b>50.92±3.52</b>
	Micro-F1	51.32±1.52	52.42±3.52	<b>55.81±3.20</b>
Freebase	Macro-F1	27.44±1.96	28.16±2.47	<b>30.57±2.94</b>
	Micro-F1	55.14±1.33	56.15±0.34	<b>56.96±0.67</b>

Table 3: Ablation Study: w/o Graph Synthesized and Meta-Target Graph.

#### 5.4 Parameter Analysis

In this subsection, we explore the sensitivity of the hyperparameter  $k$  in NoiseHGNN. The parameter  $k$  determines the number of neighbor nodes to be used in the  $S$  high-order synthesized graph based on  $k$ -nearest neighbors (kNN). To understand its impact on our model’s performance, we search the number of neighbors  $k$  in the range of  $\{5, 15, 25, 35, 45\}$  for all datasets. Our results indicate that selecting an appropriate  $k$  value can significantly enhance the accuracy of NoiseHGNN across various datasets. As is demonstrated in Fig. 3, the best selection for each dataset is different, i.e.,  $k = 15$  for Freebase and IMDB,  $k = 25$  for Cora and PubMed\_NC, and  $k = 35$  for ACM. It is commonly observed that selecting a value of  $k$  that is either too large or too small can lead to suboptimal performance. We hypothesize that an excessively small  $k$  may restrict the inclusion of beneficial neighbors, while an overly large  $k$  might introduce noisy connections, thereby degrading the overall performance.

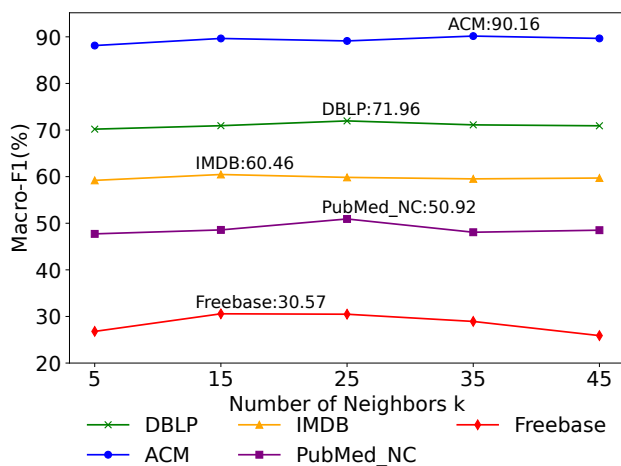


Figure 3: Effect of parameter  $k$  in top- $k(\cdot)$  selection in equation 4.

#### 5.5 Visualization Analysis

To visually represent and compare the quality of the embeddings, Fig. 4 presents the t-SNE plot (Van der Maaten and Hinton 2008) of the node embeddings generated by NoiseHGNN on the DBLP dataset. Consistent with the quantitative results, the 2D projections of the embeddings learned by NoiseHGNN show more distinguishable clusters, both visually and numerically, compared to simpleHGN and SlotGAT. The Silhouette scores support this (Rousseeuw 1987), where NoiseHGNN achieves a score of 0.21 on DBLP, significantly higher than the scores of 0.12 for simpleHGN and 0.13 for SlotGAT.

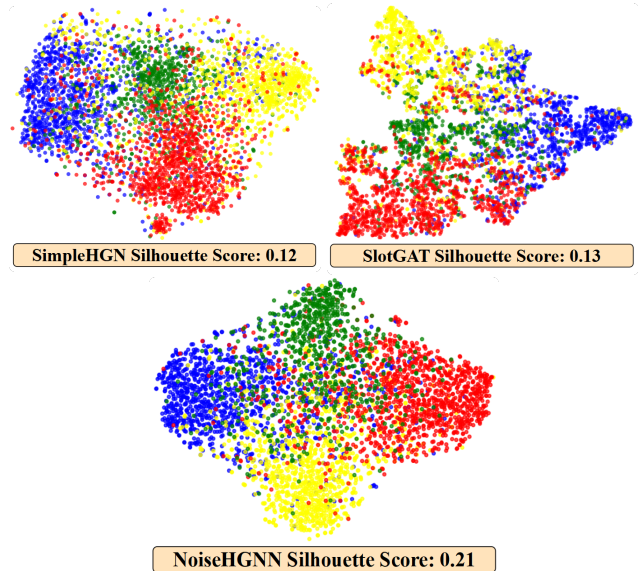


Figure 4: The t-SNE visualization of the graph representation. The proposed method achieves a good silhouette score (0.21).

### 6 Conclusion, and Future Work

This paper presents the first study addressing the problem of error link perturbation in heterogeneous graphs. To tackle this issue, we propose a novel method, **NoiseHGNN**, which effectively extracts valid structural information from the original data, thereby mitigating the negative impact of erroneous links. Extensive experiments on numerous real-world datasets demonstrate that the proposed method achieves state-of-the-art performance in noisy heterogeneous graph learning tasks. In highlights, +5~6% improvements are observed in several noised datasets compared with previous SOTA methods.

In the future, we intend to explore the performance of heterogeneous graphs in complex environments. Specifically, we intend to address the problem of model failure in the presence of missing features, missing links, and erroneous link perturbations. We hope to advance the robustness and applicability of heterogeneous graph neural networks in more complex and uncertain real-world scenarios.

## Acknowledgments

This paper is the result of the research project funded by the National Natural Foundation of China (Grant No. 62106216 and 62162064) and the Open Foundation of Yunnan Key Laboratory of Software Engineering under Grant No.2023SE104.

## References

- Davis, A. P.; Grondin, C. J.; Johnson, R. J.; Sciaky, D.; McMorran, R.; Wieggers, J.; Wieggers, T. C.; and Mattingly, C. J. 2019. The comparative toxicogenomics database: update 2019. *Nucleic acids research*, 47(D1): D948–D954.
- Dou, Y.; Liu, Z.; Sun, L.; Deng, Y.; Peng, H.; and Yu, P. S. 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.
- Du, C.; Yao, K.; Zhu, H.; Wang, D.; Zhuang, F.; and Xiong, H. 2023. Seq-HGNN: learning sequential node representation on heterogeneous graph. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1721–1730.
- Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019. Graph neural networks for social recommendation. In *The world wide web conference*, 417–426.
- Fatemi, B.; El Asri, L.; and Kazemi, S. M. 2021. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems*, 34: 22667–22681.
- Fout, A.; Byrd, J.; Shariat, B.; and Ben-Hur, A. 2017. Protein interface prediction using graph convolutional networks. *Advances in neural information processing systems*, 30.
- Franceschi, L.; Niepert, M.; Pontil, M.; and He, X. 2019. Learning discrete structures for graph neural networks. In *International conference on machine learning*, 1972–1982. PMLR.
- Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of the web conference 2020*, 2331–2341.
- Gaudelet, T.; Day, B.; Jamasb, A. R.; Soman, J.; Regep, C.; Liu, G.; Hayter, J. B.; Vickers, R.; Roberts, C.; Tang, J.; et al. 2021. Utilizing graph machine learning within drug discovery and development. *Briefings in bioinformatics*, 22(6): bbab159.
- Hu, Z.; Dong, Y.; Wang, K.; and Sun, Y. 2020. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, 2704–2710.
- Ji, H.; Zhu, J.; Shi, C.; Wang, X.; Wang, B.; Zhang, C.; Zhu, Z.; Zhang, F.; and Li, Y. 2021. Large-scale comb-k recommendation. In *Proceedings of the Web Conference 2021*, 2512–2523.
- Jin, W.; Ma, Y.; Liu, X.; Tang, X.; Wang, S.; and Tang, J. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 66–74.
- Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR*.
- Li, C.; and Goldwasser, D. 2019. Encoding social information with graph convolutional networks for political perspective detection in news media. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2594–2604.
- Liu, Y.; Zheng, Y.; Zhang, D.; Chen, H.; Peng, H.; and Pan, S. 2022. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference 2022*, 1392–1403.
- Lv, Q.; Ding, M.; Liu, Q.; Chen, Y.; Feng, W.; He, S.; Zhou, C.; Jiang, J.; Dong, Y.; and Tang, J. 2021. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 1150–1160.
- Qiu, J.; Tang, J.; Ma, H.; Dong, Y.; Wang, K.; and Tang, J. 2018. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2110–2119.
- Rousseeuw, P. J. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20: 53–65.
- Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, 593–607. Springer.
- Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.
- Sun, Y.; and Han, J. 2012. *Mining heterogeneous information networks: principles and methodologies*. Morgan & Claypool Publishers.
- Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).
- Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. In *ICLR*.
- Wang, R.; Mou, S.; Wang, X.; Xiao, W.; Ju, Q.; Shi, C.; and Xie, X. 2021. Graph structure estimation neural networks. In *Proceedings of the web conference 2021*, 342–353.
- Wang, W.; Yin, H.; Du, X.; Hua, W.; Li, Y.; and Nguyen, Q. V. H. 2019a. Online user representation learning across heterogeneous social networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 545–554.
- Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019b. Heterogeneous graph attention network. In *The world wide web conference*, 2022–2032.

- Wei, C.; Liang, J.; Liu, D.; and Wang, F. 2022. Contrastive graph structure learning via information bottleneck for recommendation. *Advances in Neural Information Processing Systems*, 35: 20407–20420.
- Yun, S.; Jeong, M.; Kim, R.; Kang, J.; and Kim, H. J. 2019. Graph transformer networks. *Advances in neural information processing systems*, 32.
- Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 793–803.
- Zhao, H.; Yao, Q.; Li, J.; Song, Y.; and Lee, D. L. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 635–644.
- Zhao, J.; Wang, X.; Shi, C.; Hu, B.; Song, G.; and Ye, Y. 2021. Heterogeneous graph structure learning for graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 4697–4705.
- Zhao, J.; Zhou, Z.; Guan, Z.; Zhao, W.; Ning, W.; Qiu, G.; and He, X. 2019. Intentgc: a scalable graph convolution framework fusing heterogeneous information for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2347–2357.
- Zhao, T.; Yang, C.; Li, Y.; Gan, Q.; Wang, Z.; Liang, F.; Zhao, H.; Shao, Y.; Wang, X.; and Shi, C. 2022. Space4hgnn: a novel, modularized and reproducible platform to evaluate heterogeneous graph neural network. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2776–2789.
- Zhou, Z.; Shi, J.; Yang, R.; Zou, Y.; and Li, Q. 2023. Slot-GAT: slot-based message passing for heterogeneous graphs. In *International Conference on Machine Learning*, 42644–42657. PMLR.
- Zhu, S.; Zhou, C.; Pan, S.; Zhu, X.; and Wang, B. 2019. Relation structure-aware heterogeneous graph neural network. In *2019 IEEE international conference on data mining (ICDM)*, 1534–1539. IEEE.