Self-supervised Heterogeneous Graph Neural Network with Co-contrastive Learning

Xiao Wang
xiaowang@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Nian Liu
nianliu@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Hui Han
hanhui@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Chuan Shi*
shichuan@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

ABSTRACT

Heterogeneous graph neural networks (HGNNs) as an emerging technique have shown superior capacity of dealing with heterogeneous information network (HIN). However, most HGNNs follow a semi-supervised learning manner, which notably limits their wide use in reality since labels are usually scarce in real applications. Recently, contrastive learning, a self-supervised method, becomes one of the most exciting learning paradigms and shows great potential when there are no labels. In this paper, we study the problem of self-supervised HGNNs and propose a novel co-contrastive learning mechanism for HGNNs, named HeCo. Different from traditional contrastive learning which only focuses on contrasting positive and negative samples, HeCo employs cross-view contrastive mechanism. Specifically, two views of a HIN (network schema and meta-path views) are proposed to learn node embeddings, so as to capture both of local and high-order structures simultaneously. Then the cross-view contrastive learning, as well as a view mask mechanism, is proposed, which is able to extract the positive and negative embeddings from two views. This enables the two views to collaboratively supervise each other and finally learn high-level node embeddings. Moreover, two extensions of HeCo are designed to generate harder negative samples with high quality, which further boosts the performance of HeCo. Extensive experiments conducted on a variety of real-world networks show the superior performance of the proposed methods over the state-of-the-arts.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Networks** → **Network algorithms**.

KEYWORDS

Heterogeneous information network, Heterogeneous graph neural network, Contrastive learning

ACM Reference Format:
Xiao Wang, Nian Liu, Hui Han, and Chuan Shi. 2018. Self-supervised Heterogeneous Graph Neural Network with Co-contrastive Learning. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In the real world, heterogeneous information network (HIN) or heterogeneous graph (HG) [30] is ubiquitous, due to the capacity of modeling various types of nodes and diverse interactions between them, such as bibliographic network [15], biomedical network [3] and so on. Recently, heterogeneous graph neural networks (HGNNs) have achieved great success in dealing with the HIN data, because they are able to effectively combine the mechanism of message passing with complex heterogeneity, so that the complex structures and rich semantics can be well captured. So far, HGNNs have significantly promoted the development of HIN analysis towards real-world applications, e.g., recommend system [6] and security system [7].

Basically, most HGNN studies belong to the semi-supervised learning paradigm, i.e., they usually design different heterogeneous message passing mechanisms to learn node embeddings, and then the learning procedure is supervised by a part of node labels. However, the requirement that some node labels have to be known beforehand is actually frequently violated, because it is very challenging or expensive to obtain labels in some real-world environments. For example, labeling an unknown gene accurately usually needs the enormous knowledge of molecular biology, which is not easy even for veteran researchers [15]. Recently, self-supervised learning, aiming to spontaneously find supervised signals from the data itself, becomes a promising solution for the setting without explicit labels [24]. Contrastive learning, as one typical technique of self-supervised learning, has attracted considerable attentions [2, 12, 13, 25, 33]. By extracting positive and negative samples in

*Corresponding author.

Woodstock '18, June 03–05, 2018, Woodstock, NY

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

data, contrastive learning aims at maximizing the similarity between positive samples while minimizing the similarity between negative samples. In this way, contrastive learning is able to learn the discriminative embeddings even without labels. Despite the wide use of contrastive learning in computer vision [2, 13] and natural language processing [4, 21], little effort has been made towards investigating the great potential on HIN.

In practice, designing heterogeneous graph neural networks with contrastive learning is non-trivial, we need to carefully consider the characteristics of HIN and contrastive learning. This requires us to address the following three fundamental problems:

(1) How to design a heterogeneous contrastive mechanism. A HIN consists of multiple types of nodes and relations, which naturally implies it possesses very complex structures. For example, meta-path, the composition of multiple relations, is usually used to capture the long-range structure in a HIN [31]. Different meta-paths represent different semantics, each of which reflects one aspect of HIN. To learn an effective node embedding which can fully encode these semantics, performing contrastive learning only on single meta-path view [26] is actually distant from sufficient. Therefore, investigating the heterogeneous cross-view contrastive mechanism is especially important for HGNNs.

(2) How to select proper views in a HIN. As mentioned before, cross-view contrastive learning is desired for HGNNs. Despite that one can extract many different views from a HIN because of the heterogeneity, one fundamental requirement is that the selected views should cover both of the local and high-order structures. Network schema, a meta template of HIN [30], reflects the direct connections between nodes, which naturally captures the local structure. By contrast, meta-path is widely used to extract the high-order structure. As a consequence, both of the network schema and meta-path structure views should be carefully considered.

(3) How to set a difficult contrastive task. It is well known that a proper contrastive task will further promote to learn a more discriminative embedding [1, 2, 32]. If two views are too similar, the supervised signal will be too weak to learn informative embedding. So we need to make the contrastive learning on these two views more complicated. For example, one strategy is to enhance the information diversity in two views, and the other is to generate harder negative samples of high quality. In short, designing a proper contrastive task is very crucial for HGNNs.

In this paper, we study the problem of self-supervised learning on HIN and propose a novel heterogeneous graph neural network with co-contrastive learning (HeCo). Specifically, different from previous contrastive learning which contrasts original network and the corrupted network, we choose network schema and meta-path structure as two views to collaboratively supervise each other. In network schema view, the node embedding is learned by aggregating information from its direct neighbors, which is able to capture the local structure. In meta-path view, the node embedding is learned by passing messages along multiple meta-paths, which aims at capturing high-order structure. In this way, we design a novel contrastive mechanism, which captures complex structures in HIN. To make contrast harder, we propose a view mask mechanism that hides different parts of network schema and meta-path, respectively, which will further enhance the diversity of two views and help extract higher-level factors from these two views. Moreover, we propose two extensions of HeCo, which generate more negative samples with high quality. Finally, we modestly adapt traditional contrastive loss to the graph data, where a node has many positive samples rather than only one, different from methods [2, 13] for CV. With the training going on, these two views are guided by each other and collaboratively optimize. The contributions of our work are summarized as follows:

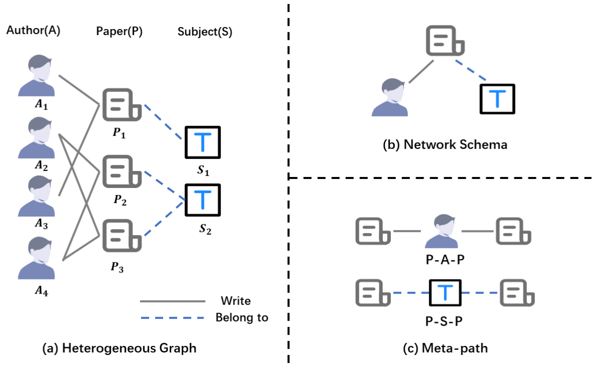
- To our best knowledge, this is the first attempt to study the self-supervised heterogeneous graph neural networks based on the cross-view contrastive learning. By contrastive learning based on cross-view manner, the high-level factors can be captured, enabling HGNNs to be better applied to real world applications without label supervision.
- We propose a novel heterogeneous graph neural network with co-contrastive learning, HeCo. HeCo innovatively employs network schema and meta-path views to collaboratively supervise each other, moreover, a view mask mechanism is designed to further enhance the contrastive performance. Additionally, two extensions of HeCo, named as HeCo_GAN and HeCo_MU, are proposed to generate negative samples with high quality.
- We conduct diverse experiments on four public datasets and the proposed HeCo outperforms the state-of-the-arts and even semi-supervised method, which demonstrates the effectiveness of HeCo from various aspects.

2 RELATED WORK

In this section,we review some closely related studies, including heterogeneous graph neural network and contrastive learning.

Heterogeneous Graph Neural Network. Graph neural networks (GNNs) have attracted considerable attentions, where most of GNNs are proposed to homogeneous graphs, and the detailed surveys can be found in [36]. Recently, some researchers focus on heterogeneous graphs. For example, HAN [35] uses hierarchical attentions to depict node-level and semantic-level structures, and on this basis, MAGNN [8] takes intermediate nodes of meta-paths into account. GTN [37] is proposed to automatically identify useful connections. HGT [16] is designed for Web-scale heterogeneous networks. In unsupervised setting, HetGNN [38] samples a fixed size of neighbors, and fuses their features using LSTMs. NSHE [40] focuses on network schema, and preserves pairwise and network schema proximity simultaneously. However, the above methods can not exploit supervised signals from data itself to learn general node embeddings.

Contrastive Learning. The approaches based on contrastive learning learn representations by contrasting positive pairs against negative pairs, and achieve great success [1, 2, 13, 25]. Here we mainly focus on reviewing the graph related contrastive learning methods. Specifically, DGI [33] builds local patches and global summary as positive pairs, and utilizes Infomax [23] theory to contrast. Along this line, GMI [27] is proposed to contrast between center node and its local patch from node features and topological structure. MVGRL [12] employs contrast across views and experiments composition between different views. GCC [28] focuses on pre-training with contrasting universally local structures from any two graphs. In heterogeneous domain, DMGI [26] conducts contrastive



(a) Heterogeneous Graph

(b) Network Schema

(c) Meta-path

Figure 1: A toy example of HIN (ACM) and relative illustrations of meta-path and network schema.

learning between original network and corrupted network on each single view, meta-path, and designs a consensus regularization to guide the fusion of different meta-paths. Nevertheless, there is a lack of methods contrasting across views in HIN so that the high-level factors can be captured.

3 PRELIMINARY

In this section, we formally define some significant concepts related to HIN as follows:

Definition 3.1. Heterogeneous Information Network. Heterogeneous Information Network (HIN) is defined as a network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \phi, \varphi)$, where $\mathcal{V}$ and $\mathcal{E}$ denote sets of nodes and edges, and it is associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and a edge type mapping function $\varphi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{A}$ and $\mathcal{R}$ denote sets of object and link types, and $|\mathcal{A} + \mathcal{R}| > 2$.

Figure 1 (a) illustrates an example of HIN. There are three types of nodes, including author (A), paper (P) and subject (S). Meanwhile, there are two types of relations ("write" and "belong to"), i.e., author writes paper, and paper belongs to subject.

Definition 3.2. Network Schema. The network schema, noted as $T_G = (\mathcal{A}, \mathcal{R})$, is a meta template for a HIN $\mathcal{G}$. Notice that T_G is a directed graph defined over object types $\mathcal{A}$, with edges as relations from $\mathcal{R}$.

For example, Figure 1 (b) is the network schema of (a), in which we can know that paper is written by author and belongs to subject. Network schema is used to describe the direct connections between different nodes, which represents local structure.

Definition 3.3. Meta-path. A meta-path $\mathcal{P}$ is defined as a path, which is in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \ldots \xrightarrow{R_l} A_{l+1}$ (abbreviated as $A_1 A_2 \ldots A_{l+1}$), which describes a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between node types A_1 and A_{l+1}, where $\circ$ denotes the composition operator on relations.

For example, Figure 1 (c) shows two meta-paths extracted from HIN in Figure 1 (a). PAP describes that two papers are written by the same author, and PSP describes that two papers belong to the same subject. Because meta-path is the combination of multiple relations, it contains complex semantics, which is regarded as high-order structure.

4 THE PROPOSED MODEL: HeCo

In this section, we propose HeCo, a novel heterogeneous graph neural network with co-contrastive learning, and the overall architecture is shown in Figure 2. Our model encodes nodes from network schema view and meta-path view, which fully captures the structures of HIN. During the encoding, we creatively involve a view mask mechanism, which makes these two views complement and supervise mutually. With the two view-specific embeddings, we employ a contrastive learning across these two views. Given the high correlation between nodes, we redefine the positive samples of a node in HIN and design a optimization strategy specially.

4.1 Node Feature Transformation

Because there are different types of nodes in a HIN, their features usually lie in different spaces. So first, we need to project features of all types of nodes into a common latent vector space, as shown in Figure 2 (a). Specifically, for a node i with type ϕ_i, we design a type-specific mapping matrix W_{ϕ_i} to transform its feature x_i into common space as follows:

$$h_i = \sigma\left(W_{\phi_i} \cdot x_i + b_{\phi_i}\right), \tag{1}$$

where $h_i \in \mathbb{R}^{d\times 1}$ is the projected feature of node i, $\sigma(\cdot)$ is an activation function, and b_{ϕ_i} denotes as vector bias, respectively.

4.2 Network Schema View Guided Encoder

Now we aim to learn the embedding of node i under network schema view, illustrated as Figure 2 (b). According to network schema, we assume that the target node i connects with S other types of nodes$\{\Phi_1, \Phi_2, \ldots, \Phi_S\}$, so the neighbors with type Φ_m of node i can be defined as $N_i^{\Phi_m}$. For node i, different types of neighbors contribute differently to its embedding, and so do the different nodes with the same type. So, we employ attention mechanism here in node-level and type-level to hierarchically aggregate messages from other types of neighbors to target node i. Specifically, we first apply node-level attention to fuse neighbors with type Φ_m:

$$h_i^{\Phi_m} = \sigma\left(\sum_{j\in N_i^{\Phi_m}} \alpha_{i,j}^{\Phi_m} \cdot h_j\right), \tag{2}$$

where σ is a nonlinear activation, h_j is the projected feature of node j, and $\alpha_{i,j}^{\Phi_m}$ denotes the attention value of node j with type Φ_m to node i. It can be calculated as follows:

$$\alpha_{i,j}^{\Phi_m} = \frac{\exp\left(LeakyReLU\left(\mathbf{a}_{\Phi_m}^{\top} \cdot [h_i || h_j]\right)\right)}{\sum\limits_{l\in N_i^{\Phi_m}} \exp\left(LeakyReLU\left(\mathbf{a}_{\Phi_m}^{\top} \cdot [h_i || h_l]\right)\right)}, \tag{3}$$

where $\mathbf{a}_{\Phi_m} \in \mathbb{R}^{2d\times 1}$ is the node-level attention vector for Φ_m and $||$ denotes concatenate operation. Please notice that in practice, we do not aggregate the information from all the neighbors in $N_i^{\Phi_m}$, but we randomly sample a part of neighbors every epoch. Specifically, if the number of neighbors with type Φ_m exceeds a predefined threshold T_{Φ_m}, we unrepeatably select T_{Φ_m} neighbors as $N_i^{\Phi_m}$, otherwise the T_{Φ_m} neighbors are selected repeatably. In this way, we ensure that every node aggregates the same amount of information from

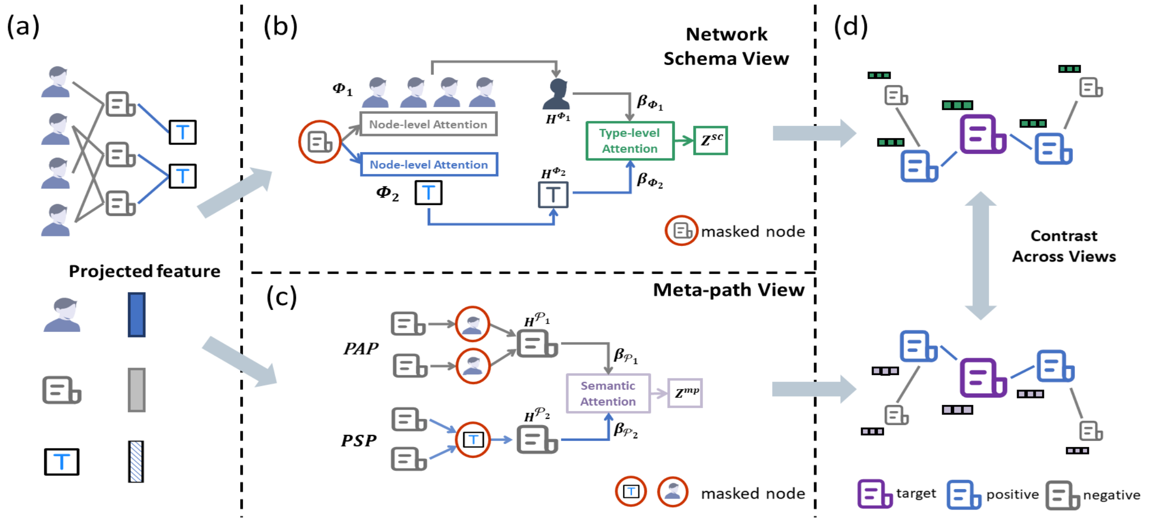


Figure 2: The overall architecture of our proposed HeCo.

neighbors, and promote the diversity of embeddings in each epoch under this view, which will make the following contrast task more challenging.

Once we get all type embeddings $\{h_i^{\Phi_1}, ..., h_i^{\Phi_S}\}$, we utilize type-level attention to fuse them together to get the final embedding z_i^{sc} for node i under network schema view. First, we measure the weight of each node type as follows:

$$\begin{aligned} w_{\Phi_m} &= \frac{1}{|V|}\sum_{i\in V} \mathbf{a}_{sc}^\top \cdot \tanh\left(\mathbf{W}_{sc} h_i^{\Phi_m} + \mathbf{b}_{sc}\right), \\ \beta_{\Phi_m} &= \frac{\exp\left(w_{\Phi_m}\right)}{\sum_{i=1}^{S} \exp\left(w_{\Phi_i}\right)}, \end{aligned} \tag{4}$$

where V is the set of target nodes, $\mathbf{W}_{sc} \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_{sc} \in \mathbb{R}^{d\times 1}$ are learnable parameters, and $\mathbf{a}_{sc}$ denotes type-level attention vector. β_{Φ_m} is interpreted as the importance of type Φ_m to target node i. So, we weighted sum the type embeddings to get z_i^{sc}:

$$z_i^{sc} = \sum_{m=1}^{S} \beta_{\Phi_m} \cdot h_i^{\Phi_m}. \tag{5}$$

4.3 Meta-path View Guided Encoder

Here we aim to learn the node embedding in the view of high-order meta-path structure, described in Figure 2 (c). Specifically, given a meta-path $\mathcal{P}_n$ from M meta-paths $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_M\}$ that start from node i, we can get the meta-path based neighbors $N_i^{\mathcal{P}_n}$. For example, as shown in Figure 1 (a), P_2 is a neighbor of P_3 based on meta-path PAP. Each meta-path represents one semantic similarity, and we apply meta-path specific GCN [20] to encode this characteristic:

$$h_i^{\mathcal{P}_n} = \frac{1}{d_i + 1} h_i + \sum_{j\in N_i^{\mathcal{P}_n}} \frac{1}{\sqrt{(d_i+1)(d_j+1)}} h_j, \tag{6}$$

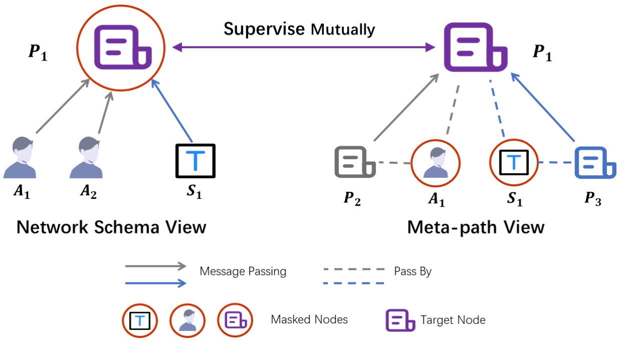


Figure 3: A schematic diagram of view mask mechanism.

where d_i and d_j are degrees of node i and j, and h_i and h_j are their projected features, respectively. With M meta-paths, we can get M embeddings $\{h_i^{\mathcal{P}_1}, ..., h_i^{\mathcal{P}_M}\}$ for node i. Then we utilize semantic-level attentions to fuse them into the final embedding z_i^{mp} under the meta-path view:

$$z_i^{mp} = \sum_{n=1}^{M} \beta_{\mathcal{P}_n} \cdot h_i^{\mathcal{P}_n}, \tag{7}$$

where $\beta_{\mathcal{P}_n}$ weighs the importance of meta-path $\mathcal{P}_n$, which is calculated as follows:

$$\begin{aligned} w_{\mathcal{P}_n} &= \frac{1}{|V|}\sum_{i\in V} \mathbf{a}_{mp}^\top \cdot \tanh\left(\mathbf{W}_{mp} h_i^{\mathcal{P}_n} + \mathbf{b}_{mp}\right), \\ \beta_{\mathcal{P}_n} &= \frac{\exp\left(w_{\mathcal{P}_n}\right)}{\sum_{i=1}^{M} \exp\left(w_{\mathcal{P}_i}\right)}, \end{aligned} \tag{8}$$

where $\mathbf{W}_{mp} \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_{mp} \in \mathbb{R}^{d\times 1}$ are the learnable parameters, and $\mathbf{a}_{mp}$ denotes the semantic-level attention vector.

4.4 View Mask Mechanism

During the generation of z_i^{sc} and z_i^{mp}, we design a view mask mechanism that hides different parts of network schema and meta-path views, respectively. In particular, we give a schematic diagram on ACM in Figure 3, where the target node is P_1. In the process of network schema encoding, P_1 only aggregates its neighbors including authors A_1, A_2 and subject S_1 into z_1^{sc}, but the message from itself is masked. While in the process of meta-path encoding, message only passes along meta-paths (e.g. PAP, PSP) from P_2 and P_3 to target P_1 to generate z_1^{mp}, while the information of intermediate nodes A_1 and S_1 are discarded. Therefore, the embeddings of node P_1 learned from these two parts are correlated but also complementary. They can supervise the training of each other, which presents a collaborative trend.

4.5 Collaboratively Contrastive Optimization

After getting the z_i^{sc} and z_i^{mp} for node i from the above two views, we feed them into a MLP with one hidden layer to map them into the space where contrastive loss is calculated:

$$\begin{aligned} z_i^{sc}\_proj &= W^{(2)}\sigma\left(W^{(1)}z_i^{sc}+b^{(1)}\right)+b^{(2)}, \\ z_i^{mp}\_proj &= W^{(2)}\sigma\left(W^{(1)}z_i^{mp}+b^{(1)}\right)+b^{(2)}, \end{aligned} \tag{9}$$

where σ is ELU non-linear function. It should be pointed out that $\{W^{(2)}, W^{(1)}, b^{(2)}, b^{(1)}\}$ are shared by two views embeddings. Next, when calculate contrastive loss, we need to define the positive and negative samples in a HIN. In computer vision, generally, one image only considers its augmentations as positive samples, and treats other images as negative samples [1, 13, 25]. In a HIN, given a node under network schema view, we can simply define its embedding learned by meta-path view as the positive sample. However, consider that the nodes are usually highly-correlated because of edges, we propose a new positive selection strategy, i.e., if two nodes are connected by many meta-paths, they are positive samples, as shown in Figure 2 (d) where links between papers represent that they are positive samples of each other. One advantage of such strategy is that the selected positive samples can well reflect the local structure of the target node.

For node i and j, we first define a function $\mathbb{C}_i(\cdot)$ to count the number of meta-paths connecting these two nodes:

$$\mathbb{C}_i(j) = \sum_{n=1}^{M} \mathbb{1}\left(j \in N_i^{\mathcal{P}_n}\right), \tag{10}$$

where $\mathbb{1}(\cdot)$ is the indicator function. Then we construct a set $S_i = \{j | j \in V \ and \ \mathbb{C}_i(j) \neq 0\}$ and sort it in the descending order based on the value of $\mathbb{C}_i(\cdot)$. Next we set a threshold T_{pos}, and if $|S_i| > T_{pos}$, we select first T_{pos} nodes from S_i as positive samples of i, denotes as $\mathbb{P}_i$, otherwise all nodes in S_i are retained. And we naturally treat all left nodes as negative samples of i, denotes as $\mathbb{N}_i$.

With the positive sample set $\mathbb{P}_i$ and negative sample set $\mathbb{N}_i$, we have the following contrastive loss under network schema view:

$$\mathcal{L}_i^{sc} = -\log \frac{\sum_{j\in\mathbb{P}_i} exp\left(sim\left(z_i^{sc}\_proj, z_j^{mp}\_proj\right)/\tau\right)}{\sum_{k\in\{\mathbb{P}_i \bigcup \mathbb{N}_i\}} exp\left(sim\left(z_i^{sc}\_proj, z_k^{mp}\_proj\right)/\tau\right)}, \tag{11}$$

where $sim(u, v)$ denotes the cosine similarity between two vectors u and v, and τ denotes a temperature parameter. We can see that different from traditional contrastive loss [2, 13], which usually only focuses on one positive pair in the numerator of eq.(11), here we consider multiple positive pairs. Also please note that for two nodes in a pair, the target embedding is from the network schema view ($z_i^{sc}\_proj$) and the embeddings of positive and negative samples are from the meta-path view ($z_k^{mp}\_proj$). In this way, we realize the cross-view self-supervision.

The contrastive loss $\mathcal{L}_i^{mp}$ is similar as $\mathcal{L}_i^{sc}$, but differently, the target embedding is from the meta-path view while the embeddings of positive and negative samples are from the network schema view. The overall objective is given as follows:

$$\mathcal{J} = \frac{1}{|V|}\sum_{i\in V}\left[\lambda\cdot\mathcal{L}_i^{sc} + (1-\lambda)\cdot\mathcal{L}_i^{mp}\right], \tag{12}$$

where λ is a coefficient to balance the effect of two views. We can optimize the proposed model via back propagation and learn the embeddings of nodes. In the end, we use z^{mp} to perform downstream tasks because nodes of target type explicitly participant into the generation of z^{mp}.

4.6 Model Extension

It is well established that a harder negative sample is very important for contrastive learning [17]. Therefore, to further improve the performance of HeCo, here we propose two extended models with new negative sample generation strategies.

HeCo_GAN GAN-based models [10, 14, 34] play a minimax game between a generator and a discriminator, and aim at forcing generator to generate fake samples, which can finally fool a well-trained discriminator. In HeCo, the negative samples are the nodes in original HIN. Here, we sample additional negatives from a continuous Gaussian distribution as in [14]. Specifically, HeCo_GAN is composed of three components: the proposed HeCo, a discriminator D and a generator G. We alternatively perform the following two steps and more details are provided in the Appendix B:

(1) We utilize two view-specific embeddings to train D and G alternatively. First, we train D to identify the embeddings from two views as positives and that generated from G as negatives. Then, we train G to generate samples with high quality that fool D. The two steps above are alternated for some interactions to make D and G trained.

(2) We utilize a well-trained G to generate samples, which can be viewed as the new negative samples with high quality. Then, we continue to train HeCo with the newly generated and original negative samples for some epochs.

HeCo_MU MixUp [39] is proposed to efficiently improve results in supervised learning by adding arbitrary two samples to create a new one. MoCHi [17] introduces this strategy into contrastive learning , who mixes the hard negatives to make harder negatives. Inspired by them, we bring this strategy into HIN field for the first time. We can get cosine similarities between node i and nodes from $\mathbb{N}_i$ during calculating eq.(11), and sort them in the descending order. Then, we select first top k negative samples as the hardest negatives, and randomly add them to create new k negatives, which

Table 1: The statistics of the datasets

Dataset	Node	Relation	Meta-path
ACM	paper (P):4019 author (A):7167 subject (S):60	P-A:13407 P-S:4019	PAP PSP
DBLP	author (A):4057 paper (P):14328 conference (C):20 term (T):7723	P-A:19645 P-C:14328 P-T:85810	APA APCPA APTPA
Freebase	movie (M):3492 actor (A):33401 direct (D):2502 writer (W):4459	M-A:65341 M-D:3762 M-W:6414	MAM MDM MWM
AMiner	paper (P):6564 author (A):13329 reference (R):35890	P-A:18007 P-R:58831	PAP PRP

are involved in training. It is worth mentioning that there are no learnable parameters in this version, which is very efficient.

5 EXPERIMENTS

5.1 Experimental Setup

Datasets We employ the following four real HIN datasets, where the basic information are summarized in Table 1.

- **ACM** [40]. The target nodes are papers, which are divided into three classes. For each paper, there are 3.33 authors averagely, and one subject.
- **DBLP** [8]. The target nodes are authors, which are divided into four classes. For each author, there are 4.84 papers averagely.
- **Freebase** [22]. The target nodes are movies, which are divided into three classes. For each movie, there are 18.7 actors, 1.07 directors and 1.83 writers averagely.
- **AMiner** [14]. The target nodes are papers. We extract a subset of original dataset, where papers are divided into four classes. For each paper, there are 2.74 authors and 8.96 references averagely.

Baselines We compare the proposed HeCo with three categories of baselines: unsupervised homogeneous methods { GraphSAGE [11], GAE [19], DGI [33] }, unsupervised heterogeneous methods { Mp2vec [5], HERec [29], HetGNN [38], DMGI [26] }, and a semi-supervised heterogeneous method HAN [35].

Implementation Details For random walk-based methods (i.e., Mp2vec, HERec, HetGNN), we set the number of walks per node to 40, the walk length to 100 and the window size to 5. For GraphSAGE, GAE, Mp2vec, HERec and DGI, we test all the meta-paths for them and report the best performance. In terms of other parameters, we follow the settings in their original papers.

For the proposed HeCo, we initialize parameters using Glorot initialization [9], and train the model with Adam [18]. We search on learning rate from 1e-4 to 5e-3, and tune the patience for early stopping from 5 to 50, i.e, we stop training if the contrastive loss does not decrease for patience consecutive epochs. For dropout used on attentions in two views, we test ranging from 0.1 to 0.5 with step 0.05, and τ is tuned from 0.5 to 0.9 also with step 0.05. Moreover, we only perform aggregation one time in each view. That is to say, for meta-path view we use one-layer GCN for every meta-path, and for network schema view we only consider interactions between nodes of target type and their one-hop neighbors of other types. The source code and datasets are publicly available on Github [1].

For all methods, we set the embedding dimension as 64 and randomly run 10 times and report the average results. For every dataset, we only use original attributes of target nodes, and assign one-hot id vectors to nodes of other types, if they are needed. For the reproducibility, we provide the specific parameter values in the supplement (Section A.3).

5.2 Node Classification

The learned embeddings of nodes are used to train a linear classifier. To more comprehensively evaluate our model, we choose 20, 40, 60 labeled nodes per class as training set, and select 1000 nodes as validation and 1000 as the test set respectively, for each dataset. We follow DMGI that report the test performance when the performance on validation gives the best result. We use common evaluation metrics, including Macro-F1, Micro-F1 and AUC. The results are reported in Table 2. As can be seen, the proposed HeCo generally outperforms all the other methods on all datasets and all splits, even compared with HAN, a semi-supervised method. We can also see that HeCo outperforms DMGI in most cases, while DMGI is even worse than other baselines with some settings, indicating that single-view is noisy and incomplete. So, performing contrastive learning across views is effective. Moreover, even HAN utilizes the label information, still, HeCo performs better than it in all cases. This well indicates the great potential of cross-view contrastive learning.

5.3 Node Clustering

In this task, we utilize K-means algorithm to the learned embeddings of all nodes and adopt normalized mutual information (NMI) and adjusted rand index (ARI) to assess the quality of the clustering results. To alleviate the instability due to different initial values, we repeat the process for 10 times, and report average results, shown in Table 3. Notice that, we do not compare with HAN, because it has known the labels of training set and been guided by validation. As we can see, HeCo consistently achieves the best results on all datasets, which proves the effectiveness of HeCo from another angle. Especially, HeCo gains about 10% improvements on NMI and 20% improvements on ARI on ACM, demonstrating the superiority of our model. Moreover, HeCo outperforms DMGI in all cases, further suggesting the importance of contrasting across views.

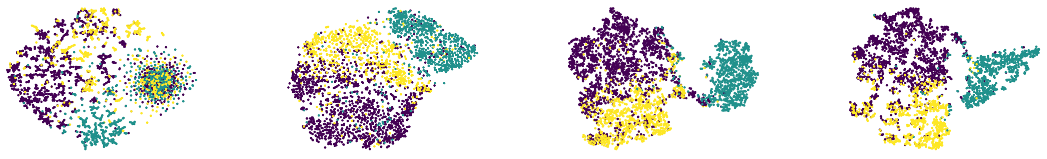
5.4 Visualization

To provide a more intuitive evaluation, we conduct embedding visualization on ACM dataset. We plot learnt embeddings of Mp2vec, DGI, DMGI and HeCo using t-SNE, and the results are shown in Figure 4, in which different colors mean different labels.

[1] https://github.com/liun-online/HeCo

Table 2: Quantitative results (%±σ) on node classification.

Datasets	Metric	Split	GraphSAGE	GAE	Mp2vec	HERec	HetGNN	HAN	DGI	DMGI	HeCo
ACM	Ma-F1	20	47.13±4.7	62.72±3.1	51.91±0.9	55.13±1.5	72.11±0.9	85.66±2.1	79.27±3.8	87.86±0.2	**88.56±0.8**
		40	55.96±6.8	61.61±3.2	62.41±0.6	61.21±0.8	72.02±0.4	87.47±1.1	80.23±3.3	86.23±0.8	**87.61±0.5**
		60	56.59±5.7	61.67±2.9	61.13±0.4	64.35±0.8	74.33±0.6	88.41±1.1	80.03±3.3	87.97±0.4	**89.04±0.5**
	Mi-F1	20	49.72±5.5	68.02±1.9	53.13±0.9	57.47±1.5	71.89±1.1	85.11±2.2	79.63±3.5	87.60±0.8	**88.13±0.8**
		40	60.98±3.5	66.38±1.9	64.43±0.6	62.62±0.9	74.46±0.8	87.21±1.2	80.41±3.0	86.02±0.9	**87.45±0.5**
		60	60.72±4.3	65.71±2.2	62.72±0.3	65.15±0.9	76.08±0.7	88.10±1.2	80.15±3.2	87.82±0.5	**88.71±0.5**
	AUC	20	65.88±3.7	79.50±2.4	71.66±0.7	75.44±1.3	84.36±1.0	93.47±1.5	91.47±2.3	**96.72±0.3**	96.49±0.3
		40	71.06±5.2	79.14±2.5	80.48±0.4	79.84±0.5	85.01±0.6	94.84±0.9	91.52±2.3	96.35±0.3	**96.40±0.4**
		60	70.45±6.2	77.90±2.8	79.33±0.4	81.64±0.7	87.64±0.7	94.68±1.4	91.41±1.9	**96.79±0.2**	96.55±0.3
DBLP	Ma-F1	20	71.97±8.4	90.90±0.1	88.98±0.2	89.57±0.4	89.51±1.1	89.31±0.9	87.93±2.4	89.94±0.4	**91.28±0.2**
		40	73.69±8.4	89.60±0.3	88.68±0.2	89.73±0.4	88.61±0.8	88.87±1.0	88.62±0.6	89.25±0.4	**90.34±0.3**
		60	73.86±8.1	90.08±0.2	90.25±0.1	90.18±0.3	89.56±0.5	89.20±0.8	89.19±0.9	89.46±0.6	**90.64±0.3**
	Mi-F1	20	71.44±8.7	91.55±0.1	89.67±0.1	90.24±0.4	90.11±1.0	90.16±0.9	88.72±2.6	90.78±0.3	**91.97±0.2**
		40	73.61±8.6	90.00±0.3	89.14±0.2	90.15±0.4	89.03±0.7	89.47±0.9	89.22±0.5	89.92±0.4	**90.76±0.3**
		60	74.05±8.3	90.95±0.2	91.17±0.1	91.01±0.3	90.43±0.6	90.34±0.8	90.35±0.8	90.66±0.5	**91.59±0.2**
	AUC	20	90.59±4.3	98.15±0.1	97.69±0.0	98.21±0.2	97.96±0.4	98.07±0.6	96.99±1.4	97.75±0.3	**98.32±0.1**
		40	91.42±4.0	97.85±0.1	97.08±0.0	97.93±0.1	97.70±0.3	97.48±0.6	97.12±0.4	97.23±0.2	**98.06±0.1**
		60	91.73±3.8	98.37±0.1	98.00±0.0	98.49±0.1	97.97±0.2	97.96±0.5	97.76±0.5	97.72±0.4	**98.59±0.1**
Freebase	Ma-F1	20	45.14±4.5	53.81±0.6	53.96±0.7	55.78±0.5	52.72±1.0	53.16±2.8	54.90±0.7	55.79±0.9	**59.23±0.7**
		40	44.88±4.1	52.44±2.3	57.80±1.1	59.28±0.6	48.57±0.5	59.63±2.3	53.40±1.4	49.88±1.9	**61.19±0.6**
		60	45.16±3.1	50.65±0.4	55.94±0.7	56.50±0.4	52.37±0.8	56.77±1.7	53.81±1.1	52.10±0.7	**60.13±1.3**
	Mi-F1	20	54.83±3.0	55.20±0.7	56.23±0.8	57.92±0.5	56.85±0.9	57.24±3.2	58.16±0.9	58.26±0.9	**61.72±0.6**
		40	57.08±3.2	56.05±2.0	61.01±1.3	62.71±0.7	53.96±1.1	63.74±2.7	57.82±0.8	54.28±1.6	**64.03±0.7**
		60	55.92±3.2	53.85±0.4	58.74±0.8	58.57±0.5	56.84±0.7	61.06±2.0	57.96±0.7	56.69±1.2	**63.61±1.6**
	AUC	20	67.63±5.0	73.03±0.7	71.78±0.7	73.89±0.4	70.84±0.7	73.26±2.1	72.80±0.6	73.19±1.2	**76.22±0.8**
		40	66.42±4.7	74.05±0.9	75.51±0.8	76.08±0.4	69.48±0.2	77.74±1.2	72.97±1.1	70.77±1.6	**78.44±0.5**
		60	66.78±3.5	71.75±0.4	74.78±0.4	74.89±0.4	71.01±0.5	75.69±1.5	73.32±0.9	73.17±1.4	**78.04±0.4**
AMiner	Ma-F1	20	42.46±2.5	60.22±2.0	54.78±0.5	58.32±1.1	50.06±0.9	56.07±3.2	51.61±3.2	59.50±2.1	**71.38±1.1**
		40	45.77±1.5	65.66±1.5	64.77±0.5	64.50±0.7	58.97±0.9	63.85±1.5	54.72±2.6	61.92±2.1	**73.75±0.5**
		60	44.91±2.0	63.74±1.6	60.65±0.3	65.53±0.7	57.34±1.4	62.02±1.2	55.45±2.4	61.15±2.5	**75.80±1.8**
	Mi-F1	20	49.68±3.1	65.78±2.9	60.82±0.4	63.64±1.1	61.49±2.5	68.86±4.6	62.39±3.9	63.93±3.3	**78.81±1.3**
		40	52.10±2.2	71.34±1.8	69.66±0.6	71.57±0.7	68.47±2.2	76.89±1.6	63.87±2.9	63.60±2.5	**80.53±0.7**
		60	51.36±2.2	67.70±1.9	63.92±0.5	69.76±0.8	65.61±2.2	74.73±1.4	63.10±3.0	62.51±2.6	**82.46±1.4**
	AUC	20	70.86±2.5	85.39±1.0	81.22±0.3	83.35±0.5	77.96±1.4	78.92±2.3	75.89±2.2	85.34±0.9	**90.82±0.6**
		40	74.44±1.3	88.29±1.0	88.82±0.2	88.70±0.4	83.14±1.6	80.72±2.1	77.86±2.1	88.02±1.3	**92.11±0.6**
		60	74.16±1.3	86.92±0.8	85.57±0.2	87.74±0.5	84.77±0.9	80.39±1.5	77.21±1.4	86.20±1.7	**92.40±0.7**



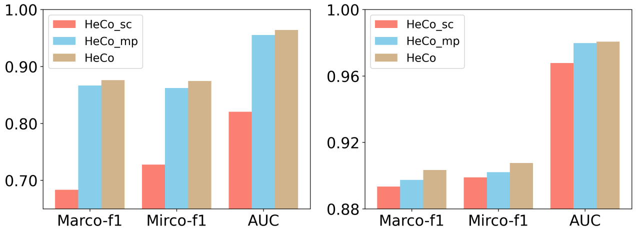
(a) Mp2vec (b) DGI (c) DMGI (d) HeCo

Figure 4: Visualization of the learned node embedding on ACM. The Silhouette scores for (a) (b) (c) (d) are 0.0292, 0.1862, 0.3015 and 0.3642, respectively.

We can see that Mp2vec and DGI present blurred boundaries between different types of nodes, because they cannot fuse all kinds of semantics. For DMGI, nodes are still mixed to some degree. The proposed HeCo correctly separates different nodes with relatively clear boundaries. Moreover, we calculate the silhouette score of different clusters, and HeCo also outperforms other three methods, demonstrating the effectiveness of HeCo again.

Table 3: Quantitative results (%±σ) on node clustering.

Datasets	ACM		DBLP		Freebase		AMiner	
Metrics	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
GraphSage	29.20	27.72	51.50	36.40	9.05	10.49	15.74	10.10
GAE	27.42	24.49	72.59	77.31	19.03	14.10	28.58	20.90
Mp2vec	48.43	34.65	73.55	77.70	16.47	17.32	30.80	25.26
HERec	47.54	35.67	70.21	73.99	19.76	19.36	27.82	20.16
HetGNN	41.53	34.81	69.79	75.34	12.25	15.01	21.46	26.60
DGI	51.73	41.16	59.23	61.85	18.34	11.29	22.06	15.93
DMGI	51.66	46.64	70.06	75.46	16.98	16.91	19.24	20.09
HeCo	**56.87**	**56.94**	**74.51**	**80.17**	**20.38**	**20.98**	**32.26**	**28.64**



(a) ACM (b) DBLP

Figure 5: The comparison of HeCo and its variants.

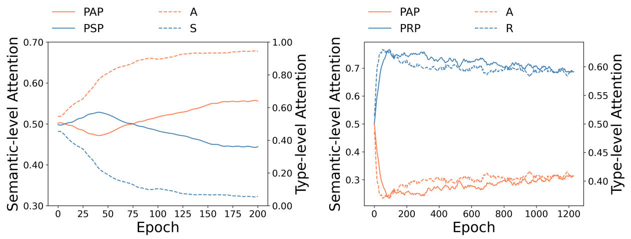
5.5 Variant Analysis

In this section, we design two variants of proposed HeCo: HeCo_sc and HeCo_mp. For variant HeCo_sc, nodes are only encoded in network schema view, and the embeddings of corresponding positive and negatives samples also come from network schema view, rather than meta-path view. For variant HeCo_mp, the practice is similar, where we only focus on meta-path view and neglect network schema view. We conduct comparison between them and HeCo on ACM and DBLP, and report the results of 40 labeled nodes per class, which are given in Figure 5.

From Figure 5, some conclusions are got as follows: (1) The results of HeCo are consistently better than two variants, indicating the effectiveness and necessity of the cross-view contrastive learning. (2) The performance of HeCo_mp is also very competitive, especially in DBLP dataset, which demonstrates that meta-path is a powerful tool to handle the heterogeneity due to capacity of capturing semantic information between nodes. (3) HeCo_sc performs worse than other methods, which makes us realize the necessity of involving the features of target nodes into embeddings if contrast is done only in a single view.

5.6 Collaborative Trend Analysis

One salient property of HeCo is the cross-view collaborative mechanism, i.e., HeCo employs the network schema and meta-path views to collaboratively supervise each other to learn the embeddings. In this section, we examine the changing trends of type-level attention β_Φ in network schema view and semantic-level attention $\beta_\mathcal{P}$ in meta-path view w.r.t epochs, and the results are plotted in Figure 6. For both ACM and AMiner, the changing trends of two views are collaborative and consistent. Specifically, for ACM, β_Φ of type A is higher than type S, and $\beta_\mathcal{P}$ of meta-path PAP also exceeds that of PSP. For AMiner, type R and meta-path PRP are more important in two views respectively. This indicates that network schema view and meta-path view adapt for each other during training and collaboratively optimize by contrasting each other.



(a) ACM (b) AMiner

Figure 6: The collaborative changing trends of attentions in two views w.r.t epochs.

5.7 Model Extension Analysis

In this section, we examine results of our extensions. As is shown above, DMGI is a rather competitive method on ACM. So, we compare our two extensions with base model and DMGI on classification and clustering tasks using ACM. The results is shown in Table 4.

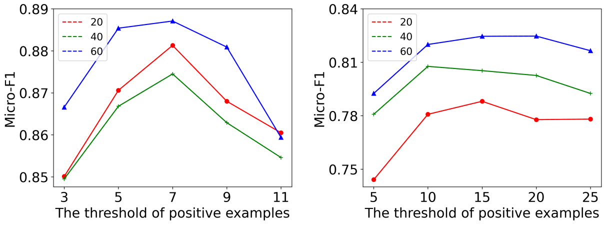
From the table, we can see that the proposed two versions generally outperform base model and DMGI, especially the version of HeCo_GAN, which improves the results with a clear margin. As expected, GAN based method can generate harder negatives that are closer to positive distributions. HeCo_MU is the second best in most cases. The better performance of HeCo_GAN and HeCo_MU indicates that more and high-quality negative samples are useful for contrastive learning in general.

Table 4: Evaluation of extended models on various tasks using ACM (Task 1: Classification; Task 2: Clustering).

Task 1		DMGI	HeCo	HeCo_MU	HeCo_GAN
Ma	20	87.86±0.2	88.56±0.8	88.65±0.8	**89.22±1.1**
	40	86.23±0.8	87.61±0.5	87.78±1.7	**88.61±1.6**
	60	87.97±0.4	89.04±0.5	**89.83±0.5**	89.55±1.3
Mi	20	87.60±0.8	88.13±0.8	88.39±0.9	**88.92±0.9**
	40	86.02±0.9	87.45±0.5	87.66±1.7	**88.48±1.7**
	60	87.82±0.5	88.71±0.5	**89.52±0.5**	89.29±1.4
AUC	20	96.72±0.3	96.49±0.3	96.38±0.5	**96.91±0.3**
	40	96.35±0.3	96.40±0.4	96.54±0.5	**97.13±0.5**
	60	96.79±0.2	96.55±0.3	96.67±0.7	**97.12±0.4**
Task 2		DMGI	HeCo	HeCo_MU	HeCo_GAN
NMI		51.66	56.87	58.17	**59.34**
ARI		46.64	56.94	59.41	**61.48**

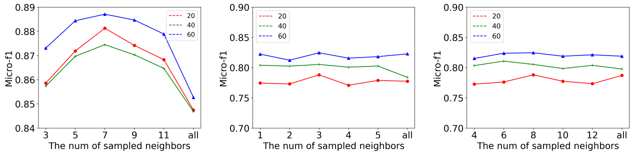
5.8 Analysis of Hyper-parameters

In this section, we systematically investigate the sensitivity of two main parameters: the threshold of positives T_{pos} and the threshold



(a) ACM (b) AMiner

Figure 7: Analysis of the threshold of positive samples.



(a) ACM: type A (b) AMiner: type A (c) AMiner: type R

Figure 8: Analysis of the number of sampled neighbors.

of sampled neighbors with T_{Φ_m}. We conduct node classification on ACM and AMiner datasets and report the Micro-F1 values.

Analysis of T_{pos}. The threshold T_{pos} determines the number of positive samples. We vary the value of it and corresponding results are shown in Figure 7. With the increase of T_{pos}, the performance goes up first and then declines, and optimum point for ACM is at 7 and at 15 for AMiner. For both datasets, three curves representing different label rates show similar changing trends.

Analysis of T_{Φ_m}. To make contrast harder, for target nodes, we randomly sample T_{Φ_m} neighbors of Φ_m type, repeatably or not. We again change the value of T_{Φ_m}. It should be pointed out that in ACM, every paper only belongs to one subject (S), so we just change the threshold of type A. The results are shown in Figure 8. As can be seen, ACM is sensitive to T_{Φ_m} of type A, and the best result is achieved when $T_{\Phi_m} = 7$. However, AMiner behaves stably with type A or type R. So in our main experiments, we set the $T_{\Phi_m} = 3$ for A and $T_{\Phi_m} = 8$ for R. Additionally, we also test the case that aggregates all neighbors without sampling, which is marked as "all" in x-axis shown in the figure. In general, "all" cannot perform very well, indicating the usefulness of sampling strategy when we aggregate neighbors.

6 CONCLUSION

In this paper, we propose a novel self-supervised heterogeneous graph neural networks with cross-view contrastive learning, named HeCo. HeCo employs network schema and meta-path as two views to capture both of local and high-order structures, and performs the contrastive learning across them. These two views are mutually supervised and finally collaboratively learn the node embeddings. Moreover, a view mask mechanism and two extensions of HeCo are designed to make the contrastive learning harder, so as to further improve the performance of HeCo. Extensive experimental results, as well as the collaboratively changing trends between these two views, verify the effectiveness of HeCo.

ACKNOWLEDGMENTS

We thank Zhenyi Wang for the useful discussion. We also thank anonymous reviewers for their time and effort in reviewing this paper. This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, 61772082, 61702296, 62002029). It is also supported by "The Fundamental Research Funds for the Central Universities 2021RC28".

REFERENCES

[1] Philip Bachman, R. Devon Hjelm, and William Buchwalter. 2019. Learning Representations by Maximizing Mutual Information Across Views. In *NeurIPS*. 15509–15519.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*. 1597–1607.

[3] Allan Peter Davis, Cynthia J. Grondin, Robin J. Johnson, Daniela Sciaky, Benjamin L. King, Roy McMorran, Jolene Wiegers, Thomas C. Wiegers, and Carolyn J. Mattingly. 2017. The Comparative Toxicogenomics Database: update 2017. *Nucleic Acids Res.* (2017), D972–D978.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[5] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *SIGKDD*. 135–144.

[6] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-guided Heterogeneous Graph Neural Network for Intent Recommendation. In *SIGKDD*. 2478–2486.

[7] Yujie Fan, Shifu Hou, Yiming Zhang, Yanfang Ye, and Melih Abdulhayoglu. 2018. Gotcha - Sly Malware!: Scorpion A Metagraph2vec Based Malware Detection System. In *SIGKDD*. 253–262.

[8] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *WWW*. 2331–2341.

[9] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.

[10] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *arXiv preprint arXiv:1406.2661* (2014).

[11] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1024–1034.

[12] Kaveh Hassani and Amir Hosein Khas Ahmadi. 2020. Contrastive Multi-View Representation Learning on Graphs. In *ICML*. 4116–4126.

[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*. 9726–9735.

[14] Binbin Hu, Yuan Fang, and Chuan Shi. 2019. Adversarial Learning on Heterogeneous Information Networks. In *SIGKDD*. 120–129.

[15] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *ICLR*.

[16] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *WWW*. 2704–2710.

[17] Yannis Kalantidis, Mert Bülent Sariyildiz, Noé Pion, Philippe Weinzaepfel, and Diane Larlus. 2020. Hard Negative Mixing for Contrastive Learning. In *NeurIPS*.

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[19] Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[20] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[21] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

[22] Xiang Li, Danhao Ding, Ben Kao, Yizhou Sun, and Nikos Mamoulis. 2020. Leveraging Meta-path Contexts for Classification in Heterogeneous Information Networks. *arXiv preprint arXiv:2012.10024* (2020).

[23] Ralph Linsker. 1988. Self-Organization in a Perceptual Network. *Computer* (1988), 105–117.

[24] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. 2020. Self-supervised learning: Generative or contrastive. *arXiv preprint arXiv:2006.08218* (2020).

[25] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[26] Chanyoung Park, Donghyun Kim, Jiawei Han, and Hwanjo Yu. 2020. Unsupervised Attributed Multiplex Network Embedding. In *AAAI*. 5371–5378.

[27] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph Representation Learning via Graphical Mutual Information Maximization. In *WWW*. 259–270.

[28] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *KDD*. 1150–1160.

[29] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. 2019. Heterogeneous Information Network Embedding for Recommendation. *IEEE Trans. Knowl. Data Eng.* (2019), 357–370.

[30] Yizhou Sun and Jiawei Han. 2012. Mining heterogeneous information networks: a structural analysis approach. *SIGKDD Explor.* (2012), 20–28.

[31] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *VLDB.* (2011), 992–1003.

[32] Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020. Contrastive Multiview Coding. In *ECCV*. 776–794.

[33] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *ICLR*.

[34] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. GraphGAN: Graph Representation Learning With Generative Adversarial Nets. In *AAAI*. 2508–2515.

[35] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.

[36] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* (2021), 4–24.

[37] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. 2019. Graph Transformer Networks. In *NeurIPS*. 11960–11970.

[38] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous Graph Neural Network. In *SIGKDD*. 793–803.

[39] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond Empirical Risk Minimization. In *ICLR*.

[40] Jianan Zhao, Xiao Wang, Chuan Shi, Zekuan Liu, and Yanfang Ye. 2020. Network Schema Preserving Heterogeneous Information Network Embedding. In *IJCAI*. 1366–1372.

A SUPPLEMENT

In the supplement, for the reproducibility, we provide all the baselines and datasets websites. The implementation details, including the detailed hyper-parameter values, are also provided.

A.1 Baselines

The publicly available implementations of baselines can be found at the following URLs:

- GraphSAGE: https://github.com/williamleif/GraphSAGE
- GAE: https://github.com/tkipf/gae
- Mp2vec: https://ericdongyx.github.io/metapath2vec/m2v.html
- HERec: https://github.com/librahu/HERec
- HetGNN: https://github.com/chuxuzhang/KDD2019_HetGNN
- HAN: https://github.com/Jhy1993/HAN
- DGI: https://github.com/PetarV-/DGI
- DMGI: https://github.com/pcy1302/DMGI

A.2 Datasets

The datasets used in experiments can be found in these URLs:

- ACM: https://github.com/Andy-Border/NSHE
- DBLP: https://github.com/cynricfu/MAGNN
- Freebase: https://github.com/dingdanhao110/Conch
- AMiner: https://github.com/librahu/HIN-Datasets-for-Recommendation-and-Network-Embedding

A.3 Implementation Details

We implement HeCo in PyTorch, and list some important parameter values used in our model in Table 5. In this table, *lr* is the learning rate, and *sample_num* is T_{Φ_m}, the threshold of sampled neighbors of type Φ_m. It should be pointed that author only connects with paper in the network schema of DBLP, so we only set the threshold for type P. *dropout_feat* is the dropout value used on projected features, and *dropout_attn* is the dropout of attentions in two views.

B DETAILS OF HeCo_GAN

In this section, we further explain the training process of HeCo_GAN, proposed in section 4.6.

As mentioned above, HeCo_GAN contains the proposed HeCo, a discriminator D and a generator G. At the beginning of the train, the parameters of HeCo should be warmed up to improve the quality of generated embeddings. So, we first only train HeCo for K_0 epochs, which is a hyper-parameter. Then, we get z^{sc} and z^{mp} and utilize them to train D and G alternatively, which is as following two steps:

- Freeze G and train D for K_D epochs. For target node i and its embedding z_i^{sc} under network schema view, we can get the embeddings of nodes in $\mathbb{P}_i$ under meta-path view. D outputs a probability that a sample j is from $\mathbb{P}_i$ given z_i^{sc}:

$$D\left(z_j | z_i^{sc}\right) = \frac{1}{1 + \exp\left(-z_i^{sc\top} M_{mp}^D z_j\right)}, \tag{13}$$

where M_{mp}^D is a matrix that projects z_i^{sc} into the space of meta-path view. And the objective function of D under the

Table 5: The values of parameter used in HeCo.

Dataset	*lr*	*patience*	*sample_num*	τ	*dropout_feat*	*dropout_attn*	*weight_decay*
ACM	0.0008	5	A:7 ; S:1	0.8	0.3	0.5	0.0
DBLP	0.0008	30	P:6	0.9	0.4	0.35	0.0
Freebase	0.001	20	D:1 ; A:18 ; W:2	0.5	0.1	0.3	0.0
AMiner	0.003	40	A:3 ; R:8	0.5	0.5	0.5	0.0

network schema view is:

$$\begin{aligned}\mathcal{L}_{i_D}^{sc} = &-\mathop{\mathbb{E}}_{j\sim\mathbb{p}_i}\log D\left(z_j^{mp}|z_i^{sc}\right)\\ &-\mathop{\mathbb{E}}_{\widetilde{z_i^{mp}}\sim G\left(z_i^{sc}\right)}\log\left(1-D\left(\widetilde{z_i^{mp}}|z_i^{sc}\right)\right),\end{aligned}\tag{14}$$

where $\mathbb{p}_i \subset \mathbb{P}_i$, which is chosen randomly, and $\widetilde{z_i^{mp}}$ is generated by generator based on z_i^{sc}. This shows that given z_i^{sc}, D aims to identify its positive samples from meta-path view as positive and samples generated by G as negative. Notice that the number of fake samples from G is the same as $|\mathbb{p}_i|$. Similarly, we can also get the objective function of D under the meta-path view $\mathcal{L}_{i_D}^{mp}$. So, we train the discriminator D by minimizing the following loss:

$$\mathcal{L}_D = \frac{1}{|B|}\sum_{i\in B}\frac{1}{2}\left(\mathcal{L}_{i_D}^{sc}+\mathcal{L}_{i_D}^{mp}\right),\tag{15}$$

where B denotes the batch of nodes that are trained in current epoch.

- Freeze D and train G for K_G epochs. G gradually improves the quality of generated samples by fooling D. Specifically, given the target i and its embedding z_i^{sc} under network schema view, G first constructs a Gaussian distribution center on i, and draws samples from it, which is related to z_i^{sc}:

$$e_i^{mp} \sim \mathcal{N}\left(z_i^{sc\top}M_{mp}^G, \sigma^2\mathbf{I}\right),\tag{16}$$

where M_{mp}^G is also a projected function to map z_i^{sc} into meta-path space, and $\sigma^2\mathbf{I}$ is covariance. We then apply one-layer MLP to enhance the expression of the fake samples:

$$\widetilde{z_i^{mp}} = G\left(z_i^{sc}\right) = \sigma\left(We_i^{mp}+b\right).\tag{17}$$

Here, σ, W and b denote non-linear activation, weight matrix and bias vector, respectively. To fool the discriminator, generator is trained under network schema view by following loss:

$$\begin{aligned}\mathcal{L}_{i_G}^{sc} &= -\mathop{\mathbb{E}}_{\widetilde{z_i^{mp}}\sim G\left(z_i^{sc}\right)}\log D(\widetilde{z_i^{mp}}|z_i^{sc}),\\ \mathcal{L}_G &= \frac{1}{|B|}\sum_{i\sim B}\frac{1}{2}\left(\mathcal{L}_{i_G}^{sc}+\mathcal{L}_{i_G}^{mp}\right).\end{aligned}\tag{18}$$

Again, $\mathcal{L}_{i_G}^{mp}$ is attained like $\mathcal{L}_{i_G}^{sc}$.

These two steps are alternated for I_{DG} times to fully train the D and G.

Once we get the well-trained G, high-quality negative samples $\widetilde{z_i^{mp}}$ and $\widetilde{z_i^{sc}}$ will be obtained, given z_i^{sc} and z_i^{mp}, respectively. And they are combined with original negative samples from meta-path view or network schema view. Finally, the extended set of negative samples is fed into HeCo to boost the training for K_H epochs.

The training processes of the proposed HeCo, discriminator D and generator G are employed iteratively until to the convergence.