Contrastive Auxiliary Learning with Structure Transformation for Heterogeneous Graphs

Wei Du[1], Hongmin Sun[1], Hang Gao[1], Gaoyang Li[2], Ying Li[1]*

[1] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, College of Computer Science and Technology, Jilin University, China
[2]School of Life Sciences and Technology, Tongji University, China
{weidu, liying}@jlu.edu.cn, {sunhm21, gaohang23}@mails.jlu.edu.cn, lgyzngc@tongji.edu.cn

Abstract

In recent years, methods based on heterogeneous graph neural networks (HGNNs) have been widely used for embedding heterogeneous graphs (HGs) due to their ability to effectively encode the rich information from HGs into low-dimensional node embeddings. Existing HGNNs focus on neighbor aggregation and semantic fusion while neglecting the HG structure and learning paradigms. However, the original data in the HG might lack node features, which may not be effectively accounted for by existing models. Additionally, exclusively relying on a single supervised learning approach may only partially leverage the invariant information in graph data. To address these challenges, we introduce the Contrastive Auxiliary Learning Model for Heterogeneous Graphs (CALHG), which combines edge perturbation and graph diffusion to enhance graph data, allowing it to capture the inherent structural information within heterogeneous graphs fully. Additionally, we employ a category-guided multi-view contrastive learning optimizing strategy, which does not rely on positive and negative samples for model training, enabling us to capture the intrinsic invariances in heterogeneous graph data. Extensive experiments and analyses on five benchmark datasets without node features and three benchmark datasets with node features validate the effectiveness and efficiency of our novel method compared with several state-of-the-art methods.

Code — https://github.com/mlcb-jlu/CALHG

Introduction

Graph Neural Networks (GNNs) are learning and representation methods (Gori, Monfardini, and Scarselli 2005; Scarselli et al. 2008) based on graph structures, and they have wide applications in fields such as recommendation systems and knowledge graphs. In recent years, many studies have shown that leveraging heterogeneous graphs can better address problems in social networks (Hamilton, Ying, and Leskovec 2017), academic networks (Zhang et al. 2019), knowledge graphs (Sun et al. 2019), and more. Heterogeneous Graph Neural Networks (HGNNs) are specialized deep learning models for heterogeneous graphs. They aim to embed heterogeneous graph nodes from high-dimensional space into low-dimensional space while preserving both structural and semantic information of the nodes (Yang et al. 2023).

Existing HGNN models mainly focus on designing effective neighbor aggregation and semantic fusion mechanisms, paying little attention to how the structure of heterogeneous graphs and the learning paradigm of the model affect performance in downstream tasks. In real-world heterogeneous graphs, a large amount of data often lacks node features. In such cases, relying solely on information learning mechanisms based on neighbor aggregation and semantic fusion while ignoring the structural learning inherent in heterogeneous graphs can significantly impact the model's performance on final downstream tasks. Furthermore, acquiring node features may be time-consuming and labor-intensive in many real-world applications involving heterogeneous graphs. Therefore, it is urgent to address how to effectively perform representation learning without node features and the specific topology of heterogeneous graphs.

Regarding graph learning paradigms, they are primarily divided into supervised learning paradigms and self-supervised learning paradigms. Supervised Graph Neural Networks (Kipf and Welling 2017; Veličković et al. 2018; Zhang et al. 2019; Hu et al. 2020b) mainly perform representation learning by simply aggregating node features and require the use of a large amount of labeled data for training, thus enhancing their performance in downstream tasks. Despite their success, these studies do not fully leverage the rich node and structure information of the graph. To address the limitations of supervised learning, a plethora of self-supervised learning methods have emerged (Kipf and Welling 2016; Veličković et al. 2019; Sun, Lin, and Zhu 2020), offering a new learning paradigm distinct from supervised learning. In self-supervised learning, the model is trained by solving a series of manually-assisted tasks (so-called pretext tasks), where the supervisory signals are automatically obtained from the data without requiring manual annotation. With carefully designed pretext tasks, self-supervised learning enables models to learn more informative representations from unlabeled data, leading to better performance (Veličković et al. 2019), generalization (Hu et al. 2020a), and robustness (You et al. 2020b) on various downstream tasks. So, given the powerful performance of self-supervised learning, leveraging the self-supervised learning paradigm to assist in supervised learning training

*Corresponding Author.

by auxiliary learning is a direction worth exploring.

To solve the above challenging issues, we propose the Contrastive Auxiliary Learning Model for Heterogeneous Graphs (CALHG) based on graph structure transformation and category-guided multi-view contrastive learning. To thoroughly utilize the intrinsic structural information in heterogeneous graphs, we present a comprehensive graph data augmentation method, which combines edge perturbation with graph diffusion techniques, ensuring a robust and effective solution. Moreover, we put forward a unique class-guided multi-view contrastive learning optimizing strategy, which overcomes the limitations of supervised learning methods by fully leveraging the rich information in the data, thereby demonstrating its superiority. This strategy does not require positive and negative samples for model training, enabling us to fully capture the intrinsic invariance in heterogeneous graphs. Thus, by integrating contrastive learning as an auxiliary training method with supervised learning, we can fully utilize label information while profoundly exploring the invariant information in the data, enriching the learning of graph representations. Fig. 1 displays the framework of the presented CALHG.

In summary, the main contributions of this paper are as follows:

- We propose a contrastive auxiliary learning model with graph structure transformation for heterogeneous graphs to improve the performance of the supervised HGNN.
- The proposed graph structure transformation method addresses the challenges by combining edge perturbation and graph diffusion techniques to enhance graph data, which more effectively captures the inherent structural information within heterogeneous graphs.
- A category-guided multi-view contrastive learning optimizing strategy is employed, avoiding the need for positive and negative samples during model training, which helps capture the intrinsic invariances of features and samples in heterogeneous graph data.
- We conduct thorough experiments and analyses on several benchmark datasets and validate significant effectiveness advantages of the proposed model over several state-of-the-art methods.

Related Work

Heterogeneous Graph Neural Network

HGNNs that aggregate information from first-order neighbors typically employ convolutional approaches tailored for different types of edges and nodes to cope with the complexity of heterogeneous graphs. HGT (Hu et al. 2020b) parameterizes the weight matrices for heterogeneous mutual attention, message passing, and propagation steps by leveraging the meta-relations in heterogeneous graphs. HGAT (Linmei et al. 2019) proposes a dual-attention mechanism that considers the importance of different adjacent nodes and the influence of different node types on the target node and aggregates information from first-order neighbor nodes. HetSANN (Hong et al. 2020) models the transitions between heterogeneous vertices by projecting them into a low-dimensional entity space and then aggregates multi-relational information from the projected neighborhood through an attention mechanism. Simple-HGN (Lv et al. 2021) proposes a heterogeneous attention mechanism based on relation-type weight matrices and embeddings and uses it to aggregate the weighted information from first-order neighbors.

HGNNs that aggregate information based on meta-paths utilize the unique meta-path information in heterogeneous graphs for aggregation operations. For example, HAN (Wang et al. 2019) first uses a node-level attention mechanism to aggregate neighbor node information under the same meta-path and then uses a semantic-level attention mechanism to fuse information from different meta-paths. Many meta-path-based methods ignore the information of intermediate nodes in the meta-paths, leading to information loss. MAGNN (Fu et al. 2020) addresses the issue by aggregating information from intermediate nodes in meta-path instances. HAGNN (Zhu et al. 2023) performs both meta-path-based intra-type aggregation and meta-path-free cross-type aggregation.

Relying solely on the information learning mechanisms based on neighbor aggregation and semantic fusion cannot adapt to the real-world heterogeneous graphs that lack node features. We propose to combine edge perturbation and graph diffusion techniques to address this challenge, so that the inherent structural information within heterogeneous graphs can be captured more effectively.

Graph Contrastive Learning

Inspired by image representation learning, various contrastive learning attempts have been applied to graph representation learning. Following Deep InfoMax (DIM) (Hjelm et al. 2019), DGI (Veličković et al. 2019) learns by maximizing the mutual information between node representations and high-level summaries of the corresponding graphs. InfoGraph (Sun et al. 2020) adopted the DIM principle for graph classification tasks. Inspired by SimCLR (Chen et al. 2020), GRACE (Zhu et al. 2020) maximizes the consistency of corresponding node representations in two augmented graph views. Similarly, GraphCL (You et al. 2020a) learns graph-level representations by maximizing the global representations of two views of a graph. Inspired by canonical correlation analysis methods, CCA-SSG (Zhang et al. 2021) introduces a non-contrastive and non-discriminative self-supervised learning objective, overcoming the drawback of previous methods that rely on negative samples.

Although graph contrastive learning has already made some progress, it lacks supervision and neglects the limited yet beneficial label information. Therefore, our work adopts a supervised learning approach combined with graph contrastive learning, effectively utilizing the label information while capturing the intrinsic invariance of graph representations through contrastive learning methods.

Method

Inspired by multi-view contrastive learning methods on homogeneous graphs, we propose a self-supervised auxiliary

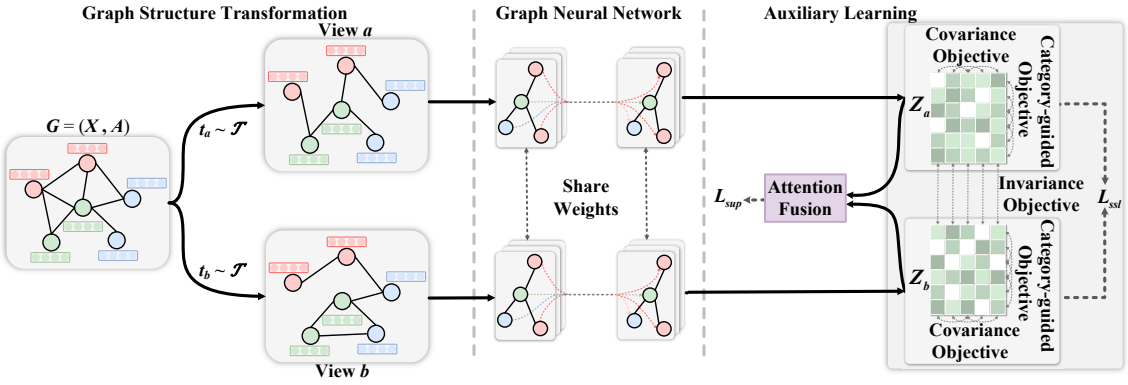

Figure 1: The overall framework of CALHG.

learning model based on graph structure transformation to enhance the performance of heterogeneous graph neural networks, as illustrated in Fig. 1. First, we homogenize the heterogeneous graph to obtain $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{A})$. Here, $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ and $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ represent the node feature matrix and adjacency matrix of the graph, respectively. N denotes the total number of nodes, and F denotes the feature dimension after feature processing. Our model consists of three components:

(1) A graph structure transformation generator $\mathcal{T}$, which defines a set of all possible graph transformation operations. $t_i \sim \mathcal{T}$ denotes a specific transformation applied to the graph $\boldsymbol{G}$. For the original graph, the transformed graph is denoted as $\widetilde{\boldsymbol{G}}_i = (\boldsymbol{X}, \widetilde{\boldsymbol{A}}_i)$.

(2) A Graph Neural Network (GNN)-based Encoder f_θ, where θ represents the parameters of the GNN encoder. All views are subsequently fed into the GNN encoder to generate node embeddings for different views: $\boldsymbol{Z}_i = f_\theta(\boldsymbol{X}, \widetilde{\boldsymbol{A}}_i)$, where $\boldsymbol{Z}_i \in \mathbb{R}^{N \times D}$ and D denotes the embedding dimension. Then, we use an attention-based mechanism for the node embeddings $\boldsymbol{Z}_i$ obtained from different views to fuse them into a fused embedding $\boldsymbol{Z}_F$.

(3) Incorporate self-supervised auxiliary learning into supervised learning models to enhance performance. Specifically, we use Invariance-Covariance Loss to constrain the distances between different feature embeddings and Category-guided Loss to constrain the distances between feature embeddings of different classes.

Graph Structure Transformation

The primary methods for graph augmentation include feature transformation and graph structure transformation. Feature transformation involves modifying the feature matrix $\boldsymbol{X}$ of the input graph data through techniques such as node attribute masking and perturbation to create new views. However, relying solely on feature transformation may pose challenges, especially when benchmark datasets lack initial node features. Additionally, research has shown that introducing noise or masking in any space can degrade model performance. Therefore, this paper utilizes graph structure transformation methods, transforming the input graph data using the adjacency matrix $\boldsymbol{A}$ to generate new views.

Edge Perturbation We utilize type-based edge dropping and k-hop-based edge generation to perturb graph edges. During actual computations, the choice of edge perturbation operation depends on the graph's density. If the graph density is greater than a threshold ϵ, we perform edge removal perturbation; if the graph density is less than ϵ, we conduct edge generation perturbation.

The process of type-based edge dropping is explicitly illustrated as follows:

$$\widetilde{\boldsymbol{A}} = \boldsymbol{A} \odot \mathbf{1}_p \tag{1}$$

where $\odot$ denotes element-wise multiplication, and $\mathbf{1}_p$ represents the perturbation location indicator matrix. For the specified type of edges to be removed, the values at the corresponding positions are set to 0, while the rest are set to 1. In addition, considering the direction of the edges, $\mathbf{1}_p$ is not necessarily a symmetric matrix.

The process of k-hop-based edge generation is explicitly illustrated as follows:

$$\widetilde{\boldsymbol{A}} = \boldsymbol{A}^k + \boldsymbol{A} \tag{2}$$

where $\boldsymbol{A}^k$ denotes the k-hop adjacency matrix of a node, and $\boldsymbol{A}$ represents the original adjacency matrix. We select a node's k-hop neighbors as its direct neighbors to compensate for the issue of insufficient graph density.

Graph Diffusion We use the graph diffusion and sparsification method for graph structure transformation. For a given heterogeneous graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, we ignore its node types and edge types, transforming it into a homogeneous graph. Then, we use the generic graph diffusion method, Personalized PageRank (PPR), to compute the connectivity between nodes in the homogeneous graph $\boldsymbol{G}$, with the specific equation as follows:

$$\boldsymbol{S} = \alpha(\boldsymbol{I}_n - (1-\alpha)\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}})^{-1} \tag{3}$$

where $\boldsymbol{S} \in \mathbb{R}^{N \times N}$ denotes the diffusion matrix, where N is the number of nodes, $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ represents the adjacency matrix of the homogeneous graph, and $\boldsymbol{D}$ is a diagonal matrix with $\boldsymbol{D}(i,i) = \sum_j \boldsymbol{A}(i,j)$. The diffusion matrix $\boldsymbol{S}$ has been proven to recover meaningful neighborhoods

from noisy graphs effectively (Gasteiger, Weißenberger, and Günnemann 2019), which is crucial for overcoming issues related to sparsity and redundancy in graph structures.

Since, during node embedding learning, a node's local neighborhood provides more information than distant neighbors, it is necessary to sparsify the dense matrix $\boldsymbol{S}$. We perform top-k sampling for each node, including the node itself, on the diffusion matrix $\boldsymbol{S}$ to obtain a subgraph $\widetilde{\boldsymbol{S}}$ for node embedding learning, with the specific equation as follows:

$$\widetilde{\boldsymbol{A}} = \mathrm{top}_{\mathrm{rank}}(\boldsymbol{S}(i,:),k) \tag{4}$$

where k controls the size of the subgraph through the number of a node's neighbors, and the $\mathrm{top}_{\mathrm{rank}}$ function returns the indices of nodes sorted by their similarity. The diffusion matrix $\boldsymbol{S}$ can be precomputed before model training starts and supports parallel computation, enabling the model to be applied to graphs that do not fit into GPU memory.

Network Architecture

Feature Preprocessing Since the input feature dimensions may vary across different node types, we use biased linear layers for each node type to map all node features into a shared feature space. The parameters in these linear layers will be optimized along with the overall model.

Encoder After obtaining different views through graph structure transformation, we use encoders to learn representations for each view. In this section, we introduce a neighbor-level attention mechanism that captures important differences among neighbors by assigning them different attention weights and aggregates neighbor information to form the node's embedding representation in a specific view. First, we employ self-attention mechanisms to learn the weights between nodes after preprocessing node features. Given a pair of neighboring nodes (i,j) in view ϕ, the neighbor-level attention e_{ij}^{ϕ} can learn how important node *j* is to node *i* in view ϕ. The importance of node pair (i,j) based on a specific view can be represented by the following:

$$e_{ij}^{\phi} = \mathrm{attn}_{\mathrm{neighbor}}(h_i^{'}, h_j^{'}, \phi) \tag{5}$$

where $\mathrm{attn}_{\mathrm{neighbor}}$ denotes the attention mechanism used to calculate the attention weights for neighbor pairs within a view. The above equation shows that, in a given view ϕ, the weight of the node pair (i,j) based on the view depends on the input node embeddings. After obtaining the importance of first-order neighbor node pairs based on the view, we normalize these using the softmax function to obtain the weight coefficients α_{ij}^{ϕ}, as shown in the follows:

$$\alpha_{ij}^{\phi} = softmax(e_{ij}^{\phi}) \tag{6}$$

Then, the view-based embedding of a node can be obtained through information aggregation of its neighbors' embeddings with the corresponding weight coefficients, as shown in the follows:

$$\boldsymbol{Z}_{\phi_i} = \sigma(\sum_{j \in N_i^{\phi}} \alpha_{ij}^{\phi} \cdot h_j^{'}) \tag{7}$$

where $\boldsymbol{Z}_{\phi_i}$ is the learnable embedding for node i in view ϕ. Given a set of views $\{\phi_1, \phi_2, \ldots, \phi_N\}$, after feeding the node features into the neighborhood-level attention module, we obtain N sets of view-specific node embeddings, which can be described as $\{\boldsymbol{Z}_{\phi_1}, \ldots, \boldsymbol{Z}_{\phi_N}\}$.

Feature Fusion Integrating the various graph structures revealed by different views is necessary to learn a more comprehensive structural representation. To address the challenge of structural information fusion in heterogeneous graphs, we utilize the view-level attention mechanism to automatically learn the importance of different view structures and fuse them for a specific task. Taking the N sets of view-specific node embeddings learned from the neighborhood-level attention module as input, we learn weights $(\beta_{\phi_1}, \beta_{\phi_2}, \ldots, \beta_{\phi_N})$ for views, as shown in the follows:

$$(\beta_{\phi_1}, \beta_{\phi_2}, \ldots, \beta_{\phi_N}) = \mathrm{att}_{\mathrm{view}}(\boldsymbol{Z}_{\phi_1}, \boldsymbol{Z}_{\phi_2}, \ldots, \boldsymbol{Z}_{\phi_N}) \tag{8}$$

where $\mathrm{att}_{\mathrm{view}}$ denotes the attention mechanism that performs view-level attention computation. Using the learned weights as coefficients, we can fuse these view-specific embeddings to obtain the final embedding $\boldsymbol{Z}_F$, as shown in the following:

$$\boldsymbol{Z}_F = \sum_{n=1}^{N} \beta_{\phi_n} \cdot \boldsymbol{Z}_{\phi_n} \tag{9}$$

Learning Objective

In this work, we use auxiliary learning to train our model and improve the performance of the supervised learning task. Formally, let Q denote the joint distribution of graph data and labels for the primary learning task, and P denote the marginal distribution of graph data. We aim to learn both a decoder f and a predictor h, where h is supervised on Q, while f is supervised on P and also trained under self-supervision. The specific learning objective of the model is as follows:

$$L = L_{sup}(f,h,Q) + \beta L_{ssl}(f,P) \tag{10}$$

where β is a positive scalar weight used to balance the two terms in the loss function.

For the supervised loss L_{sup}, we minimize the cross-entropy between the ground truth and the predictions on all labeled nodes, with the specific loss equation as follows:

$$L_{sup} = -\sum_{l \in y_L} Y^l ln(\mathrm{MLP}(Z_F^l)) \tag{11}$$

where MLP represents the multi-layer perceptron of the prediction head, y_L denotes the set of indices for labeled nodes, Y_l and Z_l^F represent the labels and the fused multi-view embeddings of the labeled nodes, respectively. Guided by labeled data, the model is optimized through backpropagation to learn the node embeddings.

Self-supervised learning loss L_{ssl} consists of Invariance-Covariance Loss and Category-guided Loss, which carry out auxiliary learning. Invariance-Covariance Loss aims to constrain the distances between different feature embeddings, while Category-guided Loss aims to constrain the distances between feature embeddings of different classes.

Invariance-Covariance Loss Inspired by methods like Barlow Twins (Zbontar et al. 2021) and VICREG (Bardes, Ponce, and LeCun 2022), which do not rely on positive and negative samples for contrastive learning, we aim to enhance the correlation of features at the same dimension across different views while reducing the correlation between different dimensions. For the original graph $\boldsymbol{G}$ transformed by $t_a \sim \mathcal{T}$ and $t_b \sim \mathcal{T}$ to get graphs $f(\boldsymbol{G}_a)$ and $f(\boldsymbol{G}_b)$, a GNN encoder with shared parameters can produce feature embeddings $\boldsymbol{Z}_a$ and $\boldsymbol{Z}_b$. We further normalize the node feature embeddings along the instance dimension so that each feature dimension has a distribution with a mean 0 and standard deviation of 1, as shown in the following:

$$\widetilde{\boldsymbol{Z}} = \frac{\boldsymbol{Z} - \mu(\boldsymbol{Z})}{\sigma(\boldsymbol{Z})} \tag{12}$$

Afterward, the normalized embeddings $\widetilde{\boldsymbol{Z}}_a$ and $\widetilde{\boldsymbol{Z}}_b$ can be used to define the invariance objective as follows:

$$i(\widetilde{\boldsymbol{Z}}_a, \widetilde{\boldsymbol{Z}}_b) = \sum_{i=j} [\widetilde{\boldsymbol{Z}}_a^T \widetilde{\boldsymbol{Z}}_b]_{i,j} \tag{13}$$

where $i = j$ indicates taking the diagonal coefficients. This term encourages the diagonal coefficients of $\widetilde{\boldsymbol{Z}}_a^T \widetilde{\boldsymbol{Z}}_b$ to be larger, increasing the similarity of the same dimensions in the feature embeddings $\boldsymbol{Z}_a$ and $\boldsymbol{Z}_b$.

Similarly, the normalized embedding $\widetilde{\boldsymbol{Z}}$ is used to define the covariance objective as follows:

$$c(\widetilde{\boldsymbol{Z}}) = \sum_{i \neq j} [\widetilde{\boldsymbol{Z}}^T \widetilde{\boldsymbol{Z}}]_{i,j}^2 \tag{14}$$

where $i \neq j$ indicates taking the off-diagonal coefficients. This term encourages the off-diagonal coefficients of $\widetilde{\boldsymbol{Z}}^T \widetilde{\boldsymbol{Z}}$ to be smaller, decorrelating the different dimensions of the embeddings and preventing them from encoding similar information.

Subsequently, the Invariance-Covariance Loss can be defined as follows:

$$L_{ic} = \frac{1}{i(\widetilde{\boldsymbol{Z}}_a, \widetilde{\boldsymbol{Z}}_b) - \alpha_1 [c(\widetilde{\boldsymbol{Z}}_a) + c(\widetilde{\boldsymbol{Z}}_b)]} \tag{15}$$

Category-guided Loss For the normalized feature embedding $\widetilde{\boldsymbol{Z}}$ obtained from any view, use the labeled data's feature embedding for category guidance to ensure that feature representations of the same category are closer and those of different categories are farther apart. If the number of categories in the labeled data is K, we can define the sample set $\boldsymbol{C} = \{\boldsymbol{C}_0, \boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_K\}$, where $\boldsymbol{C}_0$ represents unlabeled data. Then, we can define the category distance criterion as follows:

$$dis(\widetilde{\boldsymbol{Z}}_{\boldsymbol{C}_p}, \widetilde{\boldsymbol{Z}}_{\boldsymbol{C}_q}) = 1 - \frac{1}{|\boldsymbol{C}_p||\boldsymbol{C}_q|} \sum_{i \in \boldsymbol{C}_p, j \in \boldsymbol{C}_q} [\widetilde{\boldsymbol{Z}} \widetilde{\boldsymbol{Z}}^T]_{i,j} \tag{16}$$

where *i* and *j* denote samples *i* and *j*, respectively. Further, the criterion for the distance between different sample sets in the dataset can be defined as follows:

$$s(\widetilde{\boldsymbol{Z}}) = \frac{1}{K} \sum_{i=1}^{K} dis(\widetilde{\boldsymbol{Z}}_{\boldsymbol{C}_i}, \widetilde{\boldsymbol{Z}}_{\boldsymbol{C}_i}) \tag{17}$$

$$d(\widetilde{\boldsymbol{Z}}) = \frac{1}{K(K-1)} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} dis(\widetilde{\boldsymbol{Z}}_{\boldsymbol{C}_i}, \widetilde{\boldsymbol{Z}}_{\boldsymbol{C}_j}) \tag{18}$$

$$u(\widetilde{\boldsymbol{Z}}) = \frac{1}{K} \sum_{i=1}^{K} dis(\widetilde{\boldsymbol{Z}}_{\boldsymbol{C}_i}, \widetilde{\boldsymbol{Z}}_{\boldsymbol{C}_0}) \tag{19}$$

where $s(\widetilde{\boldsymbol{Z}})$ and $d(\widetilde{\boldsymbol{Z}})$ represent the average distances of feature representations within the same category and between different categories, respectively, and $u(\widetilde{\boldsymbol{Z}})$ represents the average distance from unlabeled samples to different categories in the labeled samples. For the normalized feature embeddings $\widetilde{\boldsymbol{Z}}_a$ and $\widetilde{\boldsymbol{Z}}_b$, the Category-guided Loss is defined as follows:

$$L_{cg} = \frac{s(\widetilde{\boldsymbol{Z}}_a)}{d(\widetilde{\boldsymbol{Z}}_a)} + \frac{s(\widetilde{\boldsymbol{Z}}_b)}{d(\widetilde{\boldsymbol{Z}}_b)} + \alpha_2 [u(\widetilde{\boldsymbol{Z}}_a) + u(\widetilde{\boldsymbol{Z}}_b)] \tag{20}$$

Based on Invariance-Covariance Loss and Category-guided Loss, the total self-supervised learning loss L_{ssl} is defined as follows:

$$L_{ssl} = \lambda_1 L_{ic} + \lambda_2 L_{cg} \tag{21}$$

where λ_1 and λ_2 are a positive scalar weight used to balance the two terms in the loss function.

Experiments

Experimental Settings

Datasets Experiments are conducted on five widely-used heterogeneous graphs, including ACM, DBLP, and IMDB, which contain node features, from HGB (Lv et al. 2021) and Freebase and AMiner, which do not contain node features, from HINormer (Mao et al. 2023).

Baseline Methods In addition to the proposed CALHG model, we compare it against twelve state-of-the-art incomplete multi-view clustering methods. These methods include GCN (Kipf and Welling 2017), GAT (Veličković et al. 2018), HetGNN (Zhang et al. 2019), HGT (Hu et al. 2020b), Simple-HGN (Lv et al. 2021), RGCN (Schlichtkrull et al. 2018), RSHN (Zhu et al. 2019), HAN (Wang et al. 2019), MAGNN (Fu et al. 2020), SeHGNN (Yang et al. 2023), LDMLP (Li et al. 2024), RpHGNN (Hu, Hooi, and He 2023), and HAGNN (Zhu et al. 2023).

Evaluation Each experiment is repeated 100 times to ensure robustness, and the average performance and the standard deviation are calculated. The datasets are split into training and test sets, with 80% of the labeled nodes used for training and 20% for testing. All experiments are conducted on a single GTX 4090 GPU.

Experimental Results

Table 1 displays the node classification outcomes for the eight benchmark datasets. Furthermore, Fig. 2 offers a comparison of Macro-F1 and Micro-F1 across all methods at different rates of training samples. From the results, we draw the following conclusions:

Dataset	ACM*	DBLP*	IMDB*	Freebase	AMiner	ACM	DBLP	IMDB
				Macro-F1				
GCN	90.07±1.10	89.10±0.28	14.25±0.08	59.73±4.94	75.35±1.37	91.68±0.52	87.35±1.35	56.14±1.66
GAT	91.21±1.11	92.43±0.40	38.02±2.05	58.45±5.64	75.50±1.03	92.35±0.87	93.66±0.56	57.16±2.80
HetGNN	63.60±7.49	91.77±0.48	43.91±1.11	-	-	84.17±1.10	92.11±0.49	48.40±0.89
HGT	83.38±2.81	92.43±0.99	40.80±6.74	48.03±3.09	58.90±9.94	90.33±0.70	92.28±1.04	62.13±1.60
Simple-HGN	91.70±0.82	94.17±0.48	44.80±16.69	61.40±2.45	77.01±1.40	93.27±0.49	93.57±0.52	63.97±1.40
RGCN	79.71±3.72	89.64±6.15	51.60±2.40	54.77±3.79	47.71±4.27	91.06±0.52	90.51±1.09	57.32±0.77
RSHN	74.50±4.75	80.16±2.44	44.66±1.28	59.25±0.37	38.91±0.45	88.72±1.33	87.14±0.57	51.97±0.85
HAN	87.19±0.76	91.32±0.99	47.05±1.90	46.12±1.75	55.13±5.09	90.96±0.76	92.96±0.32	60.07±1.40
MAGNN	82.24±2.96	92.86±0.42	40.47±2.67	51.61±4.41	59.04±3.90	90.87±1.16	93.19±0.52	55.73±2.51
SeHGNN	86.29±0.84	94.56±0.27	54.53±0.60	63.71±0.88	76.11±0.83	94.05±0.35	94.92±0.38	66.72±0.41
RpHGNN	90.82±0.73	93.71±0.42	60.90±0.73	59.12±1.10	65.26±2.68	92.47±0.97	94.90±0.25	**66.97±0.64**
HAGNN	92.03±1.65	94.46±0.21	26.64±16.88	64.55±3.38	76.98±6.68	92.54±0.55	94.41±0.28	64.43±1.21
CALHG	**92.50±0.40**	**95.25±0.18**	**61.15±0.37**	**65.11±0.69**	**79.21±1.11**	**94.49±0.18**	**95.17±0.35**	66.54±0.39
				Micro-F1				
GCN	90.00±1.13	89.79±0.27	40.10±0.13	68.45±1.08	85.56±0.56	91.65±0.51	88.56±1.31	63.47±0.88
GAT	91.14±1.10	92.97±0.38	52.72±0.80	66.56±0.85	85.74±0.53	92.29±0.88	94.10±0.53	63.84±1.48
HetGNN	63.58±6.69	92.31±0.46	46.91±0.99	-	-	84.03±1.09	92.63±0.45	51.28±0.84
HGT	83.31±2.80	93.10±0.81	53.82±3.90	64.29±1.56	79.45±4.44	90.27±0.68	93.00±0.85	66.90±0.84
Simple-HGN	91.58±0.83	94.58±0.45	55.36±8.41	67.38±0.91	86.52±0.54	93.19±0.50	94.06±0.48	67.71±0.82
RGCN	79.60±3.61	90.37±5.79	55.35±2.04	56.33±4.43	65.21±4.82	90.92±0.52	91.11±1.07	60.85±0.65
RSHN	74.21±5.14	80.58±2.58	48.10±0.78	63.63±0.23	57.26±0.73	88.62±1.36	87.65±0.65	56.39±0.63
HAN	87.00±0.76	92.00±0.99	55.12±1.90	61.47±2.77	74.86±3.26	90.86±0.76	93.43±0.31	65.92±0.73
MAGNN	82.00±2.97	93.40±0.38	53.27±1.69	64.63±1.54	75.55±3.00	90.77±1.15	93.81±0.49	64.21±1.43
SeHGNN	86.08±0.85	94.93±0.26	56.33±0.57	67.41±1.04	83.39±0.97	93.98±0.36	95.28±0.36	68.43±0.39
RpHGNN	90.69±0.75	94.18±0.38	62.65±0.69	63.35±1.03	78.33±3.34	92.37±1.00	95.26±0.23	**69.16±0.61**
HAGNN	91.96±1.68	94.85±0.19	45.26±9.88	69.14±1.02	86.81±2.33	92.47±0.56	94.80±0.26	67.83±0.65
CALHG	**92.42±0.40**	**95.62±0.16**	**62.97±0.34**	**70.20±0.45**	**87.05±0.65**	**94.42±0.18**	**95.55±0.32**	68.65±0.35

Table 1: Macro-F1 (%) and Micro-F1 (%) for the node classification on different datasets (mean in percentage ± standard deviation). The best and second-best results are shown in bold and underlined, respectively. '*' means using one-hot instead of the original node features. '-' means out of GPU memory.

			ACM		DBLP		IMDB		ACM*		DBLP*		IMDB*	
EP	GD	CL	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
×	×	×	92.49±0.80	92.43±0.78	91.86±0.35	92.39±0.33	64.23±1.21	67.11±0.72	82.93±17.00	84.27±13.45	91.99±0.31	92.32±0.32	57.55±1.46	60.85±0.68
×	✓	×	92.51±0.44	92.44±0.45	92.12±0.37	92.61±0.36	66.25±0.41	68.49±0.36	89.37±0.61	89.28±0.62	94.16±0.40	94.56±0.39	58.46±0.76	61.07±0.56
✓	×	×	93.74±0.27	93.66±0.27	94.42±0.37	94.82±0.33	65.21±0.86	67.97±0.60	89.80±0.40	89.64±0.41	94.70±0.57	95.09±0.52	58.61±2.47	61.63±0.58
✓	✓	×	94.14±0.25	94.06±0.25	94.57±0.50	94.97±0.36	66.48±0.39	68.64±0.32	91.50±0.41	91.38±0.42	94.57±0.39	94.97±0.36	59.15±0.73	61.71±1.35
✓	✓	✓	**94.49±0.18**	**94.42±0.18**	**95.17±0.35**	**95.55±0.32**	**66.54±0.40**	**68.65±0.35**	**92.50±0.40**	**92.43±0.40**	**95.25±0.18**	**95.62±0.16**	**61.16±0.37**	**62.97±0.34**

Table 2: Experimental results of CALHG with different terms. EP, GD, and CL denote edge perturbation, graph diffusion, and category-guided contrastive learning, respectively. '*' means using one-hot instead of the original node features.

1. Our proposed CALHG model generally outperforms existing HGNN methods on all datasets without node features. It is worth noting that our model without meta-paths, performs significantly better than the representative method Simple-HGN, which does not rely on meta-paths. In terms of Macro-F1 and Micro-F1, CALHG outperforms Simple-HGN by margins of 0.8%, 1.08%, 16.35%, 3.71%, 2.2%, and 0.85%, 1.04%, 7.61%, 2.82%, 0.53%, respectively. When compared to traditional meta-path-based methods in recent years (HAN, MAGNN, SeHGNN), our model also generally achieves superior performance. This indicates that even in the absence of node features, CALHG can still effectively learn representations using the inherent structure of the heterogeneous graph.

2. Furthermore, on most datasets with node features, our proposed CALHG model generally outperforms existing HGNN methods. Its performance is better than that of the representative method Simple-HGN, which does not rely on meta-paths. In terms of Macro-F1 and Micro-F1, CALHG surpasses Simple-HGN on the ACM, DBLP, and IMDB datasets by margins of 1.22%, 1.6%, 2.57%, and 1.23%, 1.49%, 0.94%, respectively. In addition, compared to other meta-paths-based methods, our model exhibits more stability due to its independence from the meta-paths selection. This demonstrates the superiority of our proposed CALHG in utilizing structural information and leveraging node information for graph data.

3. As depicted in Fig. 2, most HGNN methods exhibit more significant performance fluctuations with reducing train-

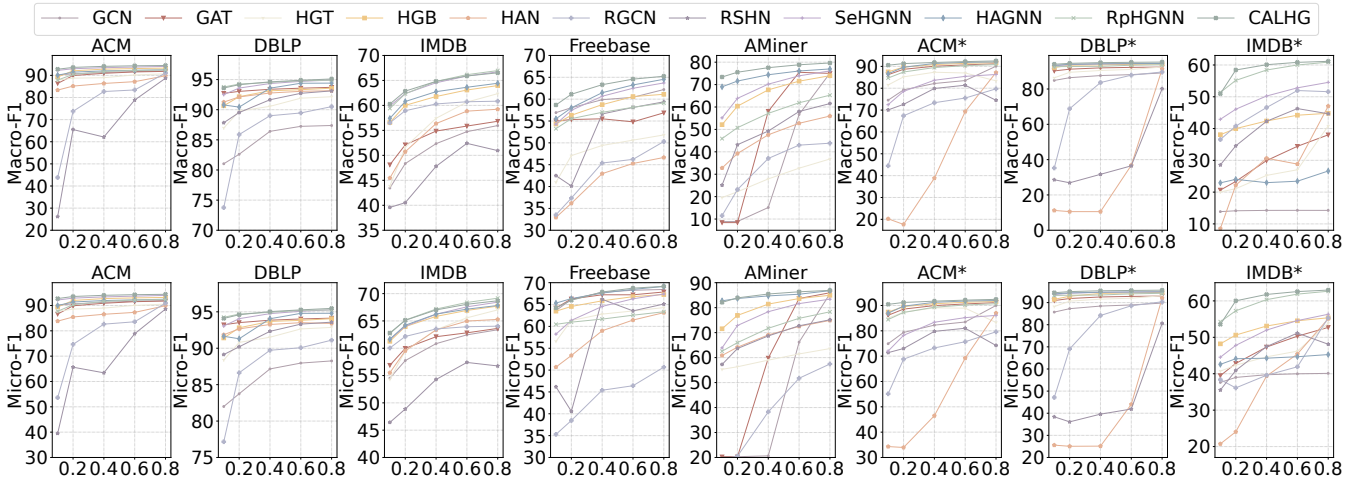

Figure 2: Macro-F1 and Micro-F1 for node classification on benchmark datasets with different rates of training samples.

ing samples, while CALHG maintains more stability. This indicates that CALHG is not significantly affected by the scale of the training sample and demonstrates good stability. This trend suggests that category-guided multi-view contrastive learning helps to capture the intrinsic invariances of features and samples in heterogeneous graph data.

Ablation Study

In order to demonstrate the effectiveness of the proposed terms, we conduct experiments by removing these terms in CALHG. As illustrated in Table 2, edge perturbation, graph diffusion, and category-guided contrastive learning, all play a significant role in the final performance of CALHG, regardless of whether the dataset includes node features.

Hyperparameter Analysis

In this section, we analyze the impact of the hyperparameters λ_1 and λ_2. The experimental results for these two hyperparameters are shown in Fig. 3. λ_1 and λ_2 control the importance of the invariance-covariance loss and the category-guided loss, respectively. We fix one of these hyperparameters and vary the other within the range of 0.1 to 1.0, showing the Macro-F1 and Micro-F1 performance of the CALHG model. As the hyperparameters change, the performance of CALHG fluctuates within a small range, indicating that it is not very sensitive to these hyperparameters.

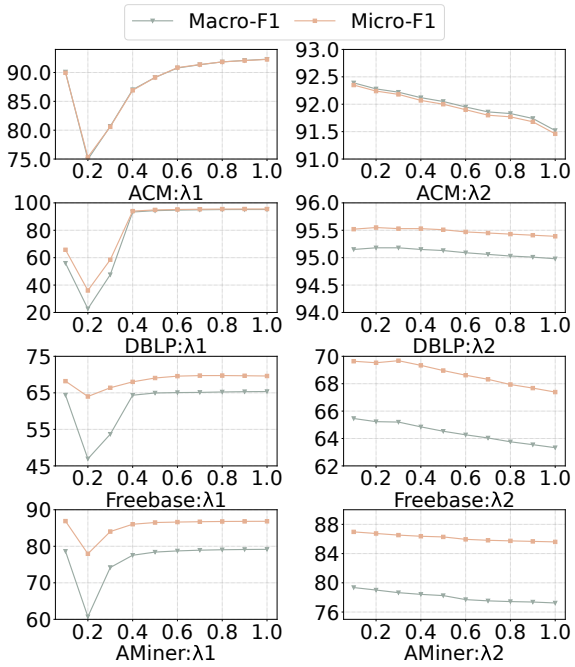

Figure 3: The hyper-parameter sensitivity of CALHG with varying λ_1 and λ_2.

Conclusion

In this paper, we propose a novel auxiliary contrastive learning model. This model combines edge perturbation and graph diffusion for graph augmentation, enabling effective learning of graph structures. Additionally, it introduces a category-guided graph contrastive learning approach, allowing for effective contrastive learning across multiple augmented graphs. We demonstrate that our method can effectively perform heterogeneous graph structure learning through extensive experiments on datasets with and without node features.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (62372494) and the Jilin Provincial Department of Science and Technology Project (No. 20240302086GX).

References

Bardes, A.; Ponce, J.; and LeCun, Y. 2022. VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning. In *International Conference on Learning Representations*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, 1597–1607.

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*, 2331–2341.

Gasteiger, J.; Weißenberger, S.; and Günnemann, S. 2019. Diffusion improves graph learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 13366–13378.

Gori, M.; Monfardini, G.; and Scarselli, F. 2005. A new model for learning in graph domains. In *Proceedings IEEE International Joint Conference on Neural Networks*, 729–734.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035.

Hjelm, R. D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Bachman, P.; Trischler, A.; and Bengio, Y. 2019. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*.

Hong, H.; Guo, H.; Lin, Y.; Yang, X.; Li, Z.; and Ye, J. 2020. An attention-based graph neural network for heterogeneous structural learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4132–4139.

Hu, J.; Hooi, B.; and He, B. 2023. Efficient Heterogeneous Graph Learning via Random Projection. *IEEE Transactions on Knowledge and Data Engineering*, 36: 8093–8107.

Hu, Z.; Dong, Y.; Wang, K.; Chang, K.-W.; and Sun, Y. 2020a. GPT-GNN: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1857–1867.

Hu, Z.; Dong, Y.; Wang, K.; and Sun, Y. 2020b. Heterogeneous graph transformer. In *Proceedings of The Web Conference*, 2704–2710.

Kipf, T.; and Welling, M. 2016. Variational Graph Auto-Encoders. *ArXiv*, abs/1611.07308.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Li, C.; Guo, Z.; He, Q.; and He, K. 2024. Long-range Meta-path Search on Large-scale Heterogeneous Graphs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Linmei, H.; Yang, T.; Shi, C.; Ji, H.; and Li, X. 2019. Heterogeneous graph attention networks for semi-supervised short text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 4821–4830.

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.

Mao, Q.; Liu, Z.; Liu, C.; and Sun, J. 2023. Hinormer: Representation learning on heterogeneous information networks with graph transformer. In *Proceedings of the ACM Web Conference 2023*, 599–610.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1): 61–80.

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 *15th International Semantic Web Conference*, 593–607.

Sun, F.-Y.; Hoffman, J.; Verma, V.; and Tang, J. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *International Conference on Learning Representations*.

Sun, K.; Lin, Z.; and Zhu, Z. 2020. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 5892–5899.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *International Conference on Learning Representations*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *International Conference on Learning Representations*.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*, 2022–2032.

Yang, X.; Yan, M.; Pan, S.; Ye, X.; and Fan, D. 2023. Simple and efficient heterogeneous graph neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 10816–10824.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020a. Graph contrastive learning with augmentations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 5812–5823.

You, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020b. When does self-supervision help graph convolutional networks? In *Proceedings of the 37th International Conference on Machine Learning*, 10871–10880.

Zbontar, J.; Jing, L.; Misra, I.; LeCun, Y.; and Deny, S. 2021. Barlow twins: Self-supervised learning via redundancy reduction. In *Proceedings of the 38th International Conference on Machine Learning*, 12310–12320.

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.

Zhang, H.; Wu, Q.; Yan, J.; Wipf, D.; and Yu, P. S. 2021. From canonical correlation analysis to self-supervised graph neural networks. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 76–89.

Zhu, G.; Zhu, Z.; Chen, H.; Yuan, C.; and Huang, Y. 2023. HAGNN: Hybrid Aggregation for Heterogeneous Graph Neural Networks. *arXiv preprint arXiv:2307.01636*.

Zhu, S.; Zhou, C.; Pan, S.; Zhu, X.; and Wang, B. 2019. Relation structure-aware heterogeneous graph neural network. In *IEEE International Conference on Data Mining (ICDM)*, 1534–1539.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*.