Critical Node-aware Augmentation for Hypergraph Contrastive Learning

Zhuo Li[1†] , **Yuena Lin**[1,3†] , **Yipeng Wang**[1] , **Wenmao Liu**[2] , **Mingliang Yu**[4] , **Zhen Yang**[1]
Gengyu Lyu[1*]
[1]Beijing University of Technology
[2]NSFOCUS Technologies Group Company Ltd.
[3]Idealism Beijing Technology Co., Ltd.
[4]Travelsky
zhuoli@emails.bjut.edu.cn, yuenalin@126.com, wangyipeng@bjut.edu.cn, liuwenmao@nsfocus.com,
164944167@qq.com, yangzhen@bjut.edu.cn, lyugengyu@gmail.com

Abstract

Hypergraph contrastive learning enables effective representation learning for hypergraphs without requiring labels. However, existing methods typically rely on randomly deleting or replacing nodes during hypergraph augmentation, which may lead to the absence of critical nodes and further disrupt the higher-order structural relationships within augmented hypergraphs. To address this issue, we propose a *Critical Node*-aware hypergraph contrastive learning method, which is the first attempt to leverage hyperedge prediction to retain critical nodes and accordingly maintain the reliable higher-order structural relationships within augmented hypergraphs. Specifically, we first employ contrastive learning to align the augmented hypergraphs, and then generate hyperedge embeddings to characterize node representations and their structural correlations. During the hyperedge embedding encoding process, we introduce a hyperedge prediction discriminator to score these embeddings, which quantifies the nodes' contributions to identify the critical nodes and maintain the higher-order structural relationships within augmented hypergraphs. Compared with previous studies, our proposed method can effectively alleviate the erroneous deletion or replacement of critical nodes and steadily maintain the inherent structural relationships between original hypergraph and augmented hypergraphs, naturally guiding better hypergraph representations for downstream tasks. Extensive experiments on various tasks demonstrate that our method is significantly superior to state-of-the-art methods.

1 Introduction

As an extension of the graph structure [Xu *et al.*, 2023], hypergraph has recently attracted increasing interest in various real-world applications, including traffic flow prediction [Ren *et al.*, 2024], user identification [Han *et al.*, 2024], and drug interaction prediction [Saifuddin *et al.*, 2023]. Hyperedge, as the fundamental component of hypergraphs, associates multiple nodes to represent higher-order structural relationships, enabling hypergraphs to capture richer information compared with the traditional edges used in graphs. For example, in the research of chemical molecules, hyperedges could capture interactions among multiple atoms, reflecting their complex spatial structure relationships in a specific molecule. These complex structures are generally difficult to represent using traditional edges, but can be effectively characterized through hyperedges. To capture complex structural information within hypergraphs, hypergraph representation learning has attracted significant attention in recent years. Among these methods, hypergraph contrastive learning has emerged as a particularly prominent approach.

Hypergraph contrastive learning enables effective representation learning for hypergraphs without requiring labels, which mainly consists of three key parts: *hypergraph augmentation*, *encoder network* and *contrastive loss*. Recent studies generally focus on improving the latter two parts to obtain better representations. For example, [Zhu *et al.*, 2023] propose a cross-view contrastive mechanism in the encoder network, which separately captures higher-order and pairwise relationships in different views. By aligning the views through node-level contrast, it obtains more expressive node representations. [Qian *et al.*, 2024] enhance encoder network with a dual-level hypergraph contrastive strategy, which effectively captures local node behaviors and groupwise interactions, further improving node classification performance [Zhu *et al.*, 2024; Wan *et al.*, 2023]. In addition, some studies focus on extending the contrastive loss to capture richer information inherent in the nodes and hyperedges. For example, [Lee and Shin, 2023] propose a tri-directional contrastive framework, which captures hypergraph structural information through node-level, hyperedge-level and node-hyperedge-level contrast, improving the quality of the learned embeddings. Obviously, these methods have obtained relatively expressive node or hyperedge representations by improving the encoder networks and contrastive losses. However, during the hypergraph augmentation process, most of above methods rely on randomly deleting or replacing nodes to generate augmented hypergraphs, which may lead to the absence of critical nodes and further disrupt higher-order structural relationships within augmented hypergraphs, naturally affecting the reliability of hypergraph representations.

To address the issue, in this paper, we propose a novel *Crit-*

ical Node-aware Hypergraph Contrastive Learning method named CNHCL, which is the first attempt to utilize hyperedge prediction to identify and retain critical nodes in augmented hypergraphs and then improve the hypergraph augmentation process. Specifically, we first align the augmented hypergraphs through contrastive loss to generate the hyperedge embeddings, which capture the interactions among nodes within each augmented hyperedge and characterize their contributions to hypergraph structural relationships. When encoding the hyperedge embeddings, we introduce a hyperedge prediction discriminator to score the hyperedge embeddings for evaluating the preservation of the higher-order structural relationships within the hyperedges, where high scores indicate that the higher-order structure relationships are well preserved after the augmentation, and low scores indicate weak preservation. After the augmentation process of each training epoch, for low-score hyperedges, we retain nodes within augmented hyperedges, identifying those that could improve the scores significantly as critical nodes. When the low-score hyperedges turn into high-score hyperedges by freeing the retained nodes from augmentations, we pick out a part of the nodes from the retained nodes as critical nodes if they greatly decrease the scores by allowing themselves to be randomly deleted with hypergraph augmentations again. For other hyperedges, we do not seek the critical nodes within them until they are evaluated as low-score hyperedges after the subsequent augmentation process. The critical nodes are then retained in the subsequent augmentation process to maintain the reliable higher-order structural relationships in augmented hypergraphs. During the model training process, in order to improve the reliability of the discriminator, we design a corresponding generator to produce fake hyperedges to strengthen it, enabling it to learn richer hyperedge representations and further improve its evaluation ability. In summary, Our main contributions lie in the following aspects:

- We propose a novel *Critical Node*-aware Hypergraph Contrastive Learning method, which is the first attempt to utilize hyperedge prediction to identify and retain critical nodes, maintaining reliable higher-order structural relationships within augmented hypergraphs.
- We design a hyperedge prediction discriminator to score hyperedge embeddings, quantifying nodes' contributions to preserve the higher-order relationships in hyperedge embeddings, meanwhile with a generator producing fake hyperedges to improve its evaluation ability.
- Extensive experiments on various datasets validate that our model guides better hypergraph representations for downstream tasks and achieves significant superiority against state-of-the-art methods.

2 Related Work

2.1 Hypergraph Neural Networks

Hypergraph neural networks have received increasing attention for modeling high-order relationships, with early works focusing on extending the graph neural network to hypergraphs. [Feng *et al.*, 2019] first generalize the convolution operation to hypergraphs, enabling efficient learning of complex hypergraph structures. [Yadati *et al.*, 2019] approximate each hyperedge of the hypergraph by a set of pairwise edges and avoid explicit Laplacian construction. Building upon the early efforts, [Bai *et al.*, 2021] incorporate attention mechanisms into the hypergraph convolution framework, dynamically adjusting hyperedge weights to improve expressiveness and reduce computational cost. Recent works have introduced more general and unified frameworks for hypergraphs. [Huang and Yang, 2021] propose a unified message-passing mechanism that processes both graphs and hypergraphs in a consistent way. [Chien *et al.*, 2022] present a multiset function framework that combines Deep Sets and Set Transformers to unify hypergraph neural networks.

2.2 Hypergraph Contrastive Learning

Hypergraph contrastive learning enables effective representation learning for hypergraphs without requiring labels [Lin *et al.*, 2021; Wei *et al.*, 2022], which mainly consists of three key parts: hypergraph augmentation, encoder network and contrastive loss. Recent methods mainly focus on improving the latter two parts to obtain more reliable learning results. For example, [Wu *et al.*, 2024] propose a collaborative contrastive method to enhance the encoder network, where the node representations of graphs and hypergraphs are jointly updated to complement each other. [Zhu *et al.*, 2023] introduce multi-view mechanism, enabling the encoder network to capture information from graphs and hypergraphs. [Lee and Chae, 2024] further incorporate an attention mechanism to dynamically assign weights to different views, enabling adaptive integration of information. [Lee and Shin, 2023] propose a TriCL framework, which captures structural information through node-level, hyperedge-level and node-hyperedge contrast, improving the quality of embeddings. However, during augmentation process, many existing methods rely on randomly deleting or replacing nodes, which may lead to the absence of critical nodes and further disrupt higher-order structural relationships within augmented hypergraphs.

2.3 Hyperedge Prediction

Hyperedge prediction is a general task for identifying and validating the existence of hyperedges within a hypergraph. [Yadati *et al.*, 2020] propose Neural Hyperlink Predictor (NHP), which adapts GCNs for hyperedge prediction in hypergraphs. [Zhang *et al.*, 2020] introduce a self-attention based graph neural network, capable of predicting variable-sized heterogeneous hyperedges. [Hwang *et al.*, 2022] utilize an adversarial generative network to overcome the limitations of traditional negative sampling methods. Recently, some studies attempt to utilize hyperedge prediction to address more complex tasks rather than validating the existence of hyperedges. For example, [Wang *et al.*, 2023] propose a novel embedding model for knowledge hypergraph prediction by employing 3D convolutions to capture higher-order interactions. Considering the ability to identify higher-order relationships of the hyperedge prediction, our model incorporates hyperedge prediction into hypergraph contrastive learning, identifying and retaining the critical nodes for maintaining reliable higher-order relationships within the augmented hypergraphs.

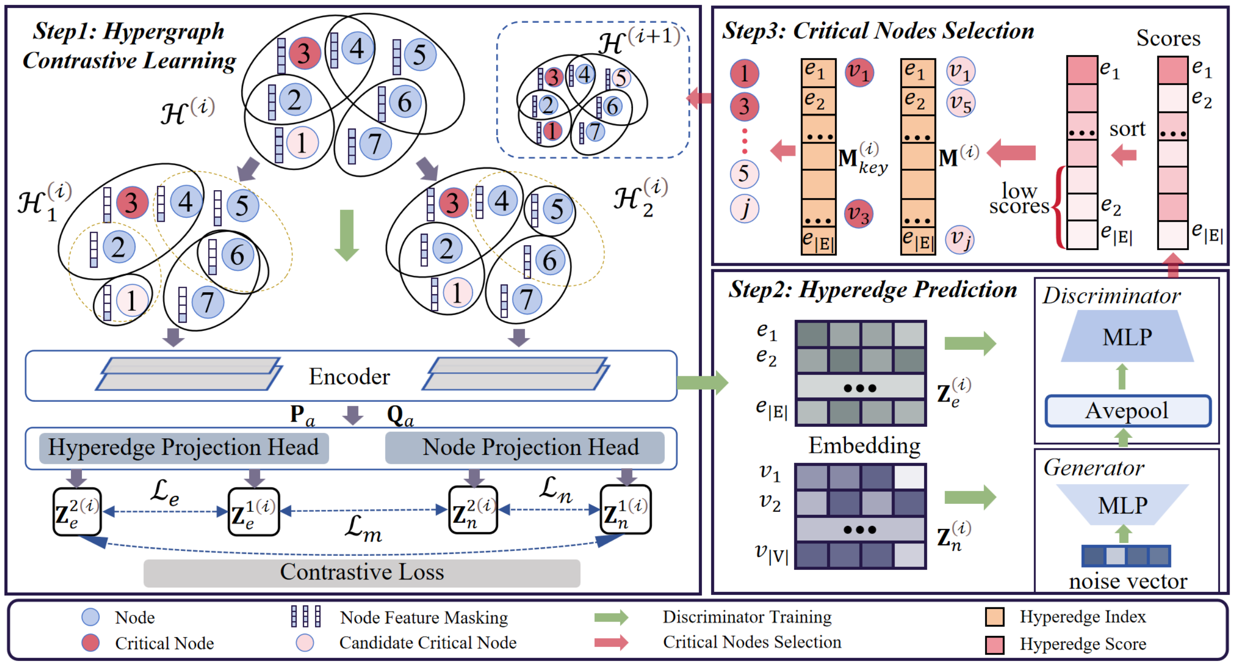

Figure 1: The overview of CNHCL. In i-th iteration, the hypergraph $\mathcal{H}^{(i)}$ are augmented by retaining both candidate critical nodes $\mathbf{M}^{(i-1)}$ and identified critical nodes $\mathbf{M}_{\text{key}}^{(i-1)}$, obtaining two augmented hypergraphs $\mathcal{H}_a^{(i)}$ $(a = 1, 2)$, which are then encoded to generate node and hyperedge embeddings $\mathbf{Z}_n^{a\,(i)}$ and $\mathbf{Z}_e^{a\,(i)}$. By aligning these embeddings, we update encoder through contrastive loss. Afterwards, the updated encoder is utilized to train *Discriminator* and *Generator*, where the *Discriminator* scores the hyperedge embeddings $\mathbf{Z}_e{}^{(i)}$ from $\mathcal{H}^{(i)}$. Finally, we sort these scores to update candidate critical nodes $\mathbf{M}^{(i)}$ and identified critical nodes $\mathbf{M}_{\text{key}}^{(i)}$, and retaining them for next iteration.

3 The Proposed Method

3.1 Overview

In this paper, we propose a novel *critical node*-aware hypergraph contrastive learning method. Previous methods mainly rely on randomly deleting or replacing nodes during hypergraph augmentation, which may lead to the absence of critical nodes and disrupt higher-order structural relationships within augmented hypergraphs. Our proposed method is the first attempt to identify and retain critical nodes when augmenting a hypergraph, and steadily maintain the inherent structural relationships within augmented hypergraphs. Specifically, as shown in Figure 1, CNHCL first utilizes the retained critical nodes (candidate critical nodes $\mathbf{M}^{(i-1)}$ and identified critical nodes $\mathbf{M}_{\text{key}}^{(i-1)}$) from the last iteration to augment current hypergraph as two augmented hypergraphs. Then, these augmented hypergraphs are encoded to generate node and hyperedge embeddings $\mathbf{Z}_n^{a\,(i)}$ and $\mathbf{Z}_e^{a\,(i)}$, and further aligned by contrastive loss to update the encoder. Afterwards, we utilize the updated encoder to train *Hyperedge Prediction Discriminator* and *Generator*, where the *Hyperedge Prediction Discriminator* scores the hyperedge embeddings $\mathbf{Z}_e{}^{(i)}$ from the original hypergraph $\mathcal{H}^{(i)}$. Finally, we sort these scores to update candidate critical nodes to be $\mathbf{M}^{(i)}$ and identified critical nodes to be $\mathbf{M}_{\text{key}}^{(i)}$, and retaining them for next iteration. During the iteration process, $\mathbf{M}^{(i)}$ represents the nodes that come from low-score hyperedges and are waiting for evaluating whether they are truly critical nodes. $\mathbf{M}_{\text{key}}^{(i)}$ represents the truly critical nodes that preserve the higher-order structural relationships in the augmented hyperedges. According to multiple iterations, CNHCL gradually identifies the truly critical nodes for hypergraph augmentation and steadily maintains the inherent structural relationships between the original hypergraph and augmented hypergraphs.

3.2 Hypergraph Contrastive Learning

Hypergraph. A general hypergraph can be presented as $\mathcal{H} = (V, E, \mathbf{X})$, where $V = \{v_1, v_2, \ldots, v_{|V|}\}$ is the set of nodes, $E = \{e_1, e_2, \ldots, e_{|E|}\}$ is the set of hyperedges, $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|V|}\}^T$ is the node feature matrix and $\boldsymbol{x}_j$ denotes the features of node v_j. To mathematically represent the hypergraph structure, we introduce an incidence matrix $\mathbf{H} \in \{0,1\}^{|V|\times|E|}$, where its element $h_{jk} = 1$ indicates node v_j belongs to hyperedge e_k, $h_{jk} = 0$ otherwise. Accordingly, the hypergraph can be represented as $\mathcal{H} = (\mathbf{H}, \mathbf{X})$. Additionally, each hyperedge $e_k \in E$ is usually assigned with a positive weight w_k to represent its importance, and all the weights formulate a diagonal matrix $\mathbf{W} \in \mathbb{R}^{|E|\times|E|}$. The degree of nodes is often defined as a diagonal matrix $\mathbf{D}_V$, where its element satisfies $d_{v_j} = \sum_k w_k h_{jk}$. Similarly, the diagonal matrix $\mathbf{D}_E$ represent the degree of hyperedges and its element satisfies $d_{e_k} = \sum_j h_{jk}$.

Hypergraph Augmentation. In i-th iteration, for the given hypergraph $\mathcal{H}^{(i)} = (\mathbf{H}^{(i)}, \mathbf{X}^{(i)})$, it consists of an incidence matrix $\mathbf{H}^{(i)}$ and a feature matrix $\mathbf{X}^{(i)}$. We augment the two parts separately and generate two augmented hypergraphs $\mathcal{H}_1^{(i)} = (\mathbf{H}_1^{(i)}, \mathbf{X}_1^{(i)})$ and $\mathcal{H}_2^{(i)} = (\mathbf{H}_2^{(i)}, \mathbf{X}_2^{(i)})$. Before conducting hypergraph augmentation, our model utilizes a hyperedge prediction discriminator (see in Section 3.3) to identify the low-score hyperedges and then select their nodes to store in candidate critical nodes $\mathbf{M}^{(i-1)}$. For high-score hyperedges, we picked out the critical nodes from $\mathbf{M}^{(i-1)}$ and store them in $\mathbf{M}_{\text{key}}^{(i-1)}$. Both $\mathbf{M}^{(i-1)}$ and $\mathbf{M}_{\text{key}}^{(i-1)}$ are implemented as dictionaries that map hyperedges to their corresponding critical nodes (see in Section 3.3).

Subsequently, we augment $\mathcal{H}^{(i)} = (\mathbf{H}^{(i)}, \mathbf{X}^{(i)})$ in i-th iteration by retaining the nodes in $\mathbf{M}^{(i-1)}$ and $\mathbf{M}_{\text{key}}^{(i-1)}$ and randomly delete from other nodes in $\mathcal{H}^{(i)}$, adjusting the node combinations within corresponding augmented hyperedges for further identifying additional critical nodes. To achieve this, we define nodes set $\mathbf{S}^{(i)}(e_k)$ of hyperedge $e_k \in E$:

$$\mathbf{S}^{(i)}(e_k) = \{v_j \in V \mid h_{jk} = 1\}. \tag{1}$$

For each hyperedge $e_k \in E$, we utilize k as its index to obtain the subsets $\mathbf{M}_k^{(i-1)}$ and $\mathbf{M}_{\text{key}:k}^{(i-1)}$ from $\mathbf{M}^{(i-1)}$ and $\mathbf{M}_{\text{key}}^{(i-1)}$, defining the critical nodes set of hyperedge $e_k \in E$ as:

$$\mathbf{S}_{\text{critical}}^{(i)}(e_k) = \mathbf{S}^{(i)}(e_k) \cap (\mathbf{M}_{\text{key}:k}^{(i-1)} \cup \mathbf{M}_k^{(i-1)}). \tag{2}$$

Then, we augment the incidence matrix $\mathbf{H}^{(i)}$ by retaining the critical nodes $\mathbf{S}_{\text{critical}}^{(i)}(e_k)$ in each hyperedge e_k, while randomly removing other nodes from $\mathbf{S}^{(i)}(e_k) \setminus \mathbf{S}_{\text{critical}}^{(i)}(e_k)$, thereby generating the augmented $\mathbf{H}_1^{(i)}$ and $\mathbf{H}_2^{(i)}$. For node features $\mathbf{X}^{(i)}$, we generate a random binary mask, where each element is sampled from a Bernoulli distribution [You *et al.*, 2020], and applied to all node features to mask certain feature dimensions, thereby generating the augmented $\mathbf{X}_1^{(i)}$ and $\mathbf{X}_2^{(i)}$. Finally, we obtain the two augmented hypergraphs $\mathcal{H}_1^{(i)} = (\mathbf{H}_1^{(i)}, \mathbf{X}_1^{(i)})$ and $\mathcal{H}_2^{(i)} = (\mathbf{H}_2^{(i)}, \mathbf{X}_2^{(i)})$.

Encoder and Projection Head. We utilize the encoder to generate node representations $\mathbf{P}_a$ and hyperedge representations $\mathbf{Q}_a$, where $a \in \{1, 2\}$ corresponds to different $\mathcal{H}_a^{(i)}$. Following the two-stage neighbor aggregation strategy [Feng *et al.*, 2019; Chien *et al.*, 2022; Lee and Shin, 2023], we update $\mathbf{P}_a$ and $\mathbf{Q}_a$ iteratively. Specifically, at the k-th layer of the encoder, the representations are updated as:

$$\begin{aligned} \mathbf{Q}_a^k &= \sigma\left(\mathbf{D}_E^{-1}\mathbf{H}^\top\mathbf{P}_a^{k-1}\mathbf{\Phi}_E^k\right), \\ \mathbf{P}_a^k &= \sigma\left(\mathbf{D}_V^{-1}\mathbf{H}\mathbf{W}\mathbf{Q}_a^k\mathbf{\Phi}_V^k\right), \end{aligned} \tag{3}$$

where $\mathbf{P}_a^0 = \mathbf{X}_a^{(i)}$, $\mathbf{\Phi}$ is the trainable weight matrix, $\mathbf{W}$ is an identity matrix, and $\sigma(\cdot)$ denotes an activation function.

[You *et al.*, 2020] demonstrate that a projection head can transform representations through non-linear mappings into a space where contrastive loss is applied, thereby improving the representation quality. Motivated by this, we introduce projection heads $\mathbf{\Psi}_n(\cdot)$ and $\mathbf{\Psi}_e(\cdot)$ to project node and hyperedge representations, respectively. Each head consists of a two-layer MLP, and maps $\mathbf{P}_a$ and $\mathbf{Q}_a$ into $\mathbf{Z}_n^{a(i)}$ and $\mathbf{Z}_e^{a(i)}$, which are then used for contrastive learning, i.e., $\mathbf{Z}_n^{a(i)} = \mathbf{\Psi}_n(\mathbf{P}_a)$ and $\mathbf{Z}_e^{a(i)} = \mathbf{\Psi}_e(\mathbf{Q}_a)$.

Hypergraph Contrastive Loss. We adopt three-level contrastive loss to align augmented hypergraphs, which consists of *Node*-level, *Hyperedge*-level, and *Hyperedge-Node*-level contrastive losses [Lee and Shin, 2023].

[*Node*-level] For each node v_j, we treat its embedding $\boldsymbol{z}_{n,j}^{1(i)}$ from $\mathbf{Z}_n^{1(i)}$ and the embedding $\boldsymbol{z}_{n,j}^{2(i)}$ from $\mathbf{Z}_n^{2(i)}$ as a positive pair since they correspond to the same node in different augmented hypergraphs. Meanwhile, we treat all other nodes embeddings in $\mathbf{Z}_n^{2(i)}$ as negative pairs for $\boldsymbol{z}_{n,j}^{1(i)}$. Then, the loss for each positive pair can be defined as:

$$l(\boldsymbol{z}_{n,j}^{1(i)}, \boldsymbol{z}_{n,j}^{2(i)}) = -\log \frac{\exp\left(s(\boldsymbol{z}_{n,j}^{1(i)}, \boldsymbol{z}_{n,j}^{2(i)})/\tau\right)}{\sum_{k\neq j}\exp\left(s(\boldsymbol{z}_{n,j}^{1(i)}, \boldsymbol{z}_{n,k}^{2(i)})/\tau\right)}, \tag{4}$$

where $s(\cdot)$ is the cosine similarity, τ is a temperature parameter. This loss encourages positive pairs to be similar while pushing negatives apart. Accordingly, the *Node*-level contrastive loss can be presented as:

$$\mathcal{L}_n^{(i)} = \frac{1}{2|V|}\sum_{j=1}^{|V|}\left[l(\boldsymbol{z}_{n,j}^{1(i)}, \boldsymbol{z}_{n,j}^{2(i)}) + l(\boldsymbol{z}_{n,j}^{2(i)}, \boldsymbol{z}_{n,j}^{1(i)})\right]. \tag{5}$$

[*Hyperedge*-level] For each hyperedge e_k, its embeddings $\boldsymbol{z}_{e,k}^{1(i)}$ in $\mathbf{Z}_e^{1(i)}$ and $\boldsymbol{z}_{e,k}^{2(i)}$ in $\mathbf{Z}_e^{2(i)}$ are treated as positive pairs. The *Hyperedge*-level contrastive loss can be presented as:

$$\mathcal{L}_e^{(i)} = \frac{1}{2|E|}\sum_{k=1}^{|E|}\left[l(\boldsymbol{z}_{e,k}^{1(i)}, \boldsymbol{z}_{e,k}^{2(i)}) + l(\boldsymbol{z}_{e,k}^{2(i)}, \boldsymbol{z}_{e,k}^{1(i)})\right]. \tag{6}$$

[*Hyperedge-Node*-level] For each node v_j, its related hyperedge e_k in the other augmented hypergraph are treated as positive pairs, while the rest are negative pairs. The *Hyperedge-Node* level contrastive loss can be presented as:

$$\mathcal{L}_m^{(i)} = \frac{1}{2|I|}\sum_{(j,k)\in I}\left[l(\boldsymbol{z}_{n,j}^{1(i)}, \boldsymbol{z}_{e,k}^{2(i)}) + l(\boldsymbol{z}_{n,j}^{2(i)}, \boldsymbol{z}_{e,k}^{1(i)})\right], \tag{7}$$

where $I = \{(j,k) \mid h_{jk} = 1\}$ denotes the set of incident node-hyperedge pairs from the incidence matrix $\mathbf{H}$. Finally, by combining Eq. (5-7), we obtain the overall hypergraph contrastive loss as follows:

$$\mathcal{L}^{(i)} = \mathcal{L}_n^{(i)} + w_e\mathcal{L}_e^{(i)} + w_m\mathcal{L}_m^{(i)}, \tag{8}$$

where w_e and w_m are the weights.

We utilize the constructed loss function $\mathcal{L}^{(i)}$ to align the two augmented hypergraphs $\mathcal{H}_1^{(i)}$ and $\mathcal{H}_2^{(i)}$, updating the encoder for the subsequent Hyperedge Prediction (Section 3.3) and Critical Nodes Selection (Section 3.4) process.

3.3 Hyperedge Prediction

In this section, we design a *Hyperedge Prediction Discriminator* and a *Generator*, which are utilized for scoring the hyperedge embeddings and selecting critical nodes in the **Critical Node Selection** stage (Section 3.4). Specifically, we first input the hypergraph $\mathcal{H}^{(i)}$ into the updated encoder to obtain node embeddings $\mathbf{Z}_n^{(i)}$ and hyperedge embeddings $\mathbf{Z}_e^{(i)}$. Then, these embeddings are used to train the *Hyperedge Prediction Discriminator* and *Generator*.

Discriminator. Given the hyperedge embeddings $\mathbf{Z}_e^{(i)}$ generated by the updated encoder, the *Discriminator* is designed to assign scores that evaluate the preservation of higher-order structural relationships within these embeddings and judge whether their contained nodes are the critical nodes. To improve the reliability of the *Discriminator*, we introduce the fake hyperedges generated from *Generator* to strengthen the *Discriminator* during the training process. Specifically, in our model, *Discriminator* consists of an average pooling layer and a multi-layer perceptron (MLP). The average pooling layer aggregates the node embeddings corresponding to the fake hyperedges to obtain their hyperedge embeddings $\mathbf{Z}_{neg}^{(i)}$, ensuring a consistent format with $\mathbf{Z}_e^{(i)}$. Then, the MLP is utilized to score the hyperedge embeddings $\boldsymbol{z}_{e,k}^{(i)}$ in $\mathbf{Z}_e^{(i)}$ and the $\boldsymbol{z}_{neg,l}^{(i)}$ in $\mathbf{Z}_{neg}^{(i)}$. We define the scoring operation of *Discriminator* as $D(\cdot)$ and apply the cross-entropy loss for training *Discriminator* following:

$$L_D^{(i)} = -\frac{1}{m}\sum_{k=1}^{m} \log D(\boldsymbol{z}_{e,k}^{(i)}) - \frac{1}{n}\sum_{l=1}^{n} \log\left(1 - D(\boldsymbol{z}_{neg,l}^{(i)})\right), \tag{9}$$

where m and n represents the number of hyperedge embeddings in $\mathbf{Z}_e^{(i)}$ and $\mathbf{Z}_{neg}^{(i)}$ respectively. The updated *Hyperedge Prediction Discriminator* will be used for the subsequent **Critical Node Selection** stage.

Generator. *Generator* plays a crucial role in improving the performance of *Discriminator* by generating challenging fake hyperedges during training, which enables *Discriminator* to learn richer hyperedge representations and further improves its evaluation capability. Specifically, in our model, *Generator* is an MLP that utilizes random noise to generate vectors for selecting nodes to form fake hyperedges [Hwang *et al.*, 2022]. *Discriminator* first utilizes fake hyperedges to update itself. Then, *Generator* generates new fake hyperedges and feeds them into average pooling to obtain fake hyperedge embeddings $\mathbf{Z'}_{neg}^{(i)}$. Moreover, $\mathbf{Z'}_{neg}^{(i)}$ are scored by the updated *Discriminator* to update *Generator*, enabling it to produce more challenging fake hyperedges in subsequent iterations. The loss function of *Generator* is presented as:

$$L_G^{(i)} = -\frac{1}{n}\sum_{l=1}^{n} \log D'(\boldsymbol{z'}_{neg,l}^{(i)}), \tag{10}$$

where $\boldsymbol{z'}_{neg,l}^{(i)} \in \mathbf{Z'}_{neg}^{(i)}$, n represents the number of embeddings in $\mathbf{Z'}_{neg}^{(i)}$, and $D'(\cdot)$ represents the scoring operation of the updated *Hyperedge Prediction Discriminator*.

3.4 Critical Nodes Selection

In this section, we first utilize the updated *Hyperedge Prediction Discriminator* to score embeddings $\boldsymbol{z}_{e,k}^{(i)} \in \mathbf{Z}_e^{(i)}$ as $D'(\boldsymbol{z}_{e,k}^{(i)})$. Then, we introduce a threshold ε to identify the low-score embeddings and store the corresponding hyperedges in a set $\mathbf{U}^{(i)}$:

$$\mathbf{U}^{(i)} = \{e_k \mid D'(\boldsymbol{z}_{e,k}^{(i)}) < \varepsilon\}, \tag{11}$$

Algorithm 1 The update process of $\mathbf{M}_{\text{key}}^{(i)}$

```
Input:
$E$: the set of hyperedges;
$\mathbf{U}^{(i)}, \mathbf{U}^{(i+1)}$: the set of low-score hyperedges;
$\mathbf{M}^{(i-1)}, \mathbf{M}^{(i)}$: candidate critical nodes;
$\mathbf{M}_{\text{key}}^{(i)}$: identified critical nodes.
Output: $\mathbf{M}_{\text{key}}^{(i+1)}$
1: for $e_k \in E$ do
2:     if $e_k \in \mathbf{U}^{(i)}$ and $e_k \notin \mathbf{U}^{(i+1)}$ then
3:         Update $\mathbf{M}_{\text{key}:k}^{(i)}$ by solving (12);
4:     else if $e_k \notin \mathbf{U}^{(i+1)}$ and $\mathbf{M}_k^{(i-1)} \neq \emptyset$ then
5:         Update $\mathbf{M}_{\text{key}:k}^{(i)}$ by solving (13);
6:     end if
7: end for
8: return $\mathbf{M}_{\text{key}}^{(i+1)}$
```

where e_k represents a low-score hyperedge, characterized by its embedding $\boldsymbol{z}_{e,k}^{(i)}$. Afterwards, we introduce the candidate critical nodes $\mathbf{M}^{(i)}$ and identified critical nodes $\mathbf{M}_{\text{key}}^{(i)}$, both of which are implemented as dictionaries that map hyperedges to their corresponding critical nodes. The nodes in $\mathbf{M}^{(i)}$ and $\mathbf{M}_{\text{key}}^{(i)}$ are retained in corresponding augmented hypergraphs in next iteration. During the process of identifying critical nodes, we consider that the low-score hyperedges are caused by excessive absence of critical nodes in augmented hyperedges. When a hyperedge e_k is identified as a low-score hyperedge, we iteratively retain the nodes from e_k into the candidate critical nodes $\mathbf{M}_k^{(i)} \subset \mathbf{M}^{(i)}$ to improve the corresponding score $D'(\boldsymbol{z}_{e,k}^{(i)})$, further evaluating these nodes' contributions through the changed scores to select critical node of e_k, which leads $D'(\boldsymbol{z}_{e,k}^{(i)}) > \varepsilon$, and store it in the identified critical nodes $\mathbf{M}_{\text{key}:k}^{(i)}$, i.e.,

$$\mathbf{M}_{\text{key}:k}^{(i+1)} = \mathbf{M}_{\text{key}:k}^{(i)} \cup (\mathbf{M}_k^{(i)} \setminus \mathbf{M}_k^{(i-1)}). \tag{12}$$

When e_k is not a low-score hyperedge, it may contain multiple candidate critical nodes while some of them may not be truly critical nodes. To identify these potential truly critical nodes, we progressively delete nodes from $\mathbf{M}_k^{(i)}$, and evaluate these nodes' contributions to select critical node of e_k, which leads $D'(\boldsymbol{z}_{e,k}^{(i)}) < \varepsilon$, then store it in $\mathbf{M}_{\text{key}:k}^{(i)}$, i.e.,

$$\mathbf{M}_{\text{key}:k}^{(i+1)} = \mathbf{M}_{\text{key}:k}^{(i)} \cup (\mathbf{M}_k^{(i-1)} \setminus \mathbf{M}_k^{(i)}). \tag{13}$$

According to the above operations, we sequentially obtain the identified critical nodes $\mathbf{M}_{\text{key}}^{(i+1)}$ of all hyperedges $\{e_k|_{k=1}^{|E|}\}$, which maintain the reliable higher-order structural relationships in augmented hypergraphs, naturally guiding expressive hypergraph representations. Algorithm 1 summarizes the update process of $\mathbf{M}_{\text{key}}^{(i)}$.

Dataset	Cora-C	Citeseer	Cora-A	Pubmed	DBLP	ModelNet40	A.R.
Node2Vec	70.99 ± 1.42	53.85 ± 1.93	58.50 ± 2.14	78.75 ± 0.94	72.09 ± 0.33	84.94 ± 0.40	12.0
GCN	77.11 ± 1.81	66.07 ± 2.45	73.66 ± 1.31	82.63 ± 0.62	87.58 ± 0.25	91.67 ± 0.22	8.6
DGI	75.89 ± 1.91	69.94 ± 1.14	79.88 ± 0.86	79.40 ± 0.63	88.00 ± 0.21	91.59 ± 0.21	7.6
HGNN	77.50 ± 1.83	66.16 ± 2.39	74.38 ± 1.22	83.42 ± 0.66	88.32 ± 0.39	92.23 ± 0.21	7.4
UniGCN	77.91 ± 1.97	66.40 ± 1.91	77.30 ± 1.46	84.08 ± 0.79	90.31 ± 0.29	94.62 ± 0.26	5.0
GRACE	78.08 ± 1.44	65.42 ± 1.31	74.11 ± 0.47	81.94 ± 0.43	89.03 ± 0.37	90.68 ± 0.34	7.6
HyperConv	75.38 ± 1.39	64.81 ± 2.34	74.86 ± 1.19	82.64 ± 0.56	87.39 ± 0.16	93.51 ± 0.12	8.0
AllSet	76.21 ± 1.77	67.83 ± 1.81	76.94 ± 1.30	82.85 ± 0.91	90.07 ± 0.30	96.85 ± 0.21	6.4
TriCL	81.57 ± 1.11	72.02 ± 1.22	82.15 ± 0.97	84.26 ± 0.68	91.12 ± 0.18	97.08 ± 0.11	3.0
VilLain	71.13 ± 1.81	65.53 ± 3.14	58.57 ± 1.34	79.87 ± 0.86	72.26 ± 0.35	86.08 ± 0.58	9.4
CNHCL-s	82.33 ± 1.26	72.31 ± 1.02	82.43 ± 0.92	84.44 ± 0.48	91.18 ± 0.08	97.18 ± 0.13	2.0
CNHCL	**82.45 ± 1.15**	**72.45 ± 0.93**	**82.52 ± 1.12**	**84.53 ± 0.45**	**91.23 ± 0.14**	**97.21 ± 0.11**	**1.0**

Table 1: Comparison results of node classification accuracy. For each dataset, the best and second-best performances are highlighted in bold and underlined, respectively. The A.R. column ranks the methods based on their average performance across all six employed datasets.

Method	C-C	CS	C-A	PB	DB	MN
Node2vec	75.31	71.10	69.11	69.51	65.33	76.26
HyperConv	79.02	**80.49**	77.21	74.34	69.92	**81.08**
TriCL	79.43	78.33	77.49	74.41	76.16	76.39
VilLain	77.30	80.08	76.79	66.84	76.27	74.15
CNHCL-s	79.79	78.69	77.93	74.57	76.21	76.66
CNHCL	**80.01**	79.68	**78.26**	**74.72**	**76.68**	76.83

Table 2: Comparison results of hyperedge prediction accuracy.

Method	C-C	CS	C-A	PB	DB	MN
Node2vec	39.93	25.62	17.18	24.13	32.79	90.16
HyperConv	38.98	34.64	35.08	22.28	56.45	90.45
TriCL	54.51	44.56	49.62	28.62	**63.10**	94.98
VilLain	40.87	21.23	13.56	**31.85**	36.23	74.74
CNHCL-s	56.26	44.63	50.04	29.20	59.58	94.79
CNHCL	**56.72**	**44.71**	**50.52**	29.54	62.30	**95.13**

Table 3: Comparison results of node embeddings clustering.

4 Experiments

4.1 Experimental Setup

Datasets. We employ six popular datasets from three categories to evaluate the performance of our proposed CNHCL method. These categories include: co-citation datasets **Cora-C (C-C)**, **Citeseer (CS)**, and **Pubmed (PB)**; co-authorship datasets **Cora-A (C-A)** and **DBLP (DB)**; computer vision and graphics dataset **ModelNet40 (MN)**.

Baselines. We evaluate our proposed CNHCL method on three classical tasks (***Node Classification***, ***Hyperedge Prediction*** and ***Clustering***) against 10 representation learning methods, including Node2vec [Grover and Leskovec, 2016], GCN [Kipf and Welling, 2017], DGI [Veličković *et al.*, 2019], HGNN [Feng *et al.*, 2019], GRACE [Zhu *et al.*, 2020], UniGCN [Huang and Yang, 2021], HyperConv [Bai *et al.*, 2021], AllSet [Chien *et al.*, 2022], TriCL [Lee and Shin, 2023] and VilLain [Lee *et al.*, 2024]. For the graph representation learning methods, we convert hypergraph data into graph structure using clique expansion and then apply these methods for comparisons. All experiments are conducted on 2 NVIDIA GeForce RTX 3090 GPUs with 24GB of memory.

4.2 Node Classification

For the task of node classification, we first randomly split up each data set into 10% for training, 10% for validation, and 80% for testing. Then, a simple linear classifier is trained and tested for evaluating all comparing methods. Table 1 reports the experimental results on six datasets, where each result is the average of 10 independent experiments.

We observe that the methods such as CNHCL and TriCL, which leverage contrastive learning on hypergraphs, consistently outperform other baseline methods. This observation suggests that the incorporation of higher-order structural relationships within augmented hypergraphs significantly enhances the quality of hypergraph representation learning. To further improve the hypergraph augmentation process, CNHCL introduces the *Hyperedge Prediction Discriminator* and the *Generator*, which effectively identify and retain critical nodes within the augmented hypergraphs. As a result, CNHCL achieves superior performance across all datasets, consistently surpassing baseline methods. Additionally, we propose a model variant CNHCL-s which retains critical nodes within a single augmented hypergraph and randomly removes the nodes in the other augmented view as [Lee and Shin, 2023] to further assess the effectiveness of our approach. Despite this constraint, CNHCL-s still outperforms all baseline methods, highlighting the importance of critical node retention in improving representation learning.

4.3 Hyperedge Prediction

For the task of hyperedge prediction, we first randomly split up each data set into 80% for training, 10% for validation, and 10% for testing. We formulate hyperedge prediction as a binary classification problem [Lee *et al.*, 2024], then, a simple linear classifier is trained and tested for evaluating recent comparing methods. Table 2 reports the experimental results on six datasets, where each result is the average of 10 independent experiments.

This task obtains hyperedge embeddings for both real and

D	G	Cora-C	Citeseer	Cora-A	Pubmed	DBLP	ModelNet40
–	–	81.57 ± 1.11	72.02 ± 1.22	82.15 ± 0.97	84.26 ± 0.68	91.12 ± 0.18	97.08 ± 0.11
✓	–	82.18 ± 1.11	72.19 ± 0.88	82.40 ± 0.64	84.35 ± 0.61	91.15 ± 0.09	97.17 ± 0.11
✓	✓	82.33 ± 1.26	72.31 ± 1.02	82.43 ± 0.92	84.44 ± 0.48	91.18 ± 0.08	97.18 ± 0.13
✓	–	82.13 ± 0.99	72.39 ± 1.25	82.36 ± 0.61	84.45 ± 0.64	91.18 ± 0.07	97.16 ± 0.11
✓	✓	**82.45 ± 1.15**	**72.45 ± 0.93**	**82.52 ± 1.12**	**84.53 ± 0.45**	**91.23 ± 0.14**	**97.21 ± 0.11**

Table 4: Ablation study on the task of node classification, including the comparison between CNHCL and its variant CNHCL-s, and the comparisons whether they use *Hyperedge Prediction Discriminator* D and the *Generator* G.

fake hyperedges via average pooling over node embeddings, which captures the higher-order structural relationships inherent in node embeddings. We observe that our proposed model CNHCL outperforms other comparing methods, which demonstrates that our model maintains the reliable higher-order structural relationships within the augmented hypergraphs. Additionally, we also see that the variant is inferior to CNHCL, the reason of which lies in that it only retains the critical nodes within one augmented hypergraph while another one is not retained. Such phenomenon further verifies the significance of preserving critical nodes.

4.4 Clustering

For the task of clustering, we evaluate the node representations by k-means clustering, where the popular Normalized Mutual Information (NMI) metric is employed for experimental comparisons. Table 3 reports the comparable results on six datasets, where each result is averaged through 10 independent experiments.

According to Table 3, we observe that CNHCL achieves excellent clustering performance on most datasets. By identifying and retaining critical nodes, CNHCL effectively captures rich local information from node representations while utilizing hyperedges to establish structural connections across different nodes. Such property allows CNHCL to naturally guide the inherent connections across different nodes and accordingly achieve more accurate node clustering results. In addition, the variant CNHCL-s also expresses the ability to capture the good local information from critical nodes, showing competitive performance on several datasets. The results validate the effectiveness of our model.

4.5 Ablation Study

We conduct an ablation study to systematically evaluate the contribution of each component in CNHCL. Specifically, we compare the model CNHCL with its variant CNHCL-s, which retains critical nodes in only one augmented view. Additionally, we remove the Hyperedge Prediction *Discriminator* (D) and *Generator* (G) from CNHCL and CNHCL-s respectively to analyze their individual effects.

Table 4 reports the comparable results on six datasets. Experimental results show that the *Discriminator* results in consistent and significant improvements in node classification performance across two models on all datasets, demonstrating its effectiveness in identifying and preserving critical nodes that contribute to more informative representations. When the Generator is further preserved, the model performance also improves, indicating its auxiliary role in strengthening the Discriminator by producing challenging fake samples that contribute to more effective training.

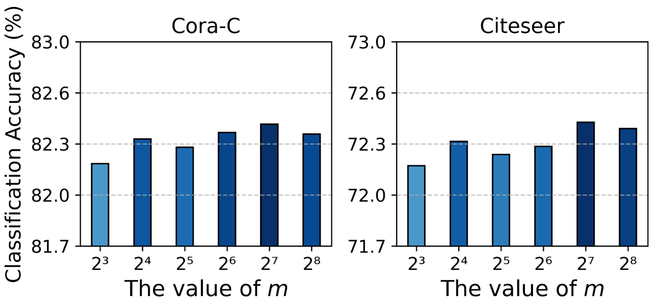

Figure 2: The impact of m on Node Classification Accuracy.

4.6 Parameter Analysis

We conduct a parameter analysis of our proposed CNHCL with respect to its employed parameter m, which is defined as the batch size of positive samples used to train the Discriminator. Figure 2 shows the node classification performance of CNHCL on the **Cora-C** and **Citeseer** datasets under different values of $m \in \{2^3, 2^4, ..., 2^8\}$. We observe that m significantly affects model performance: a small m results in insufficient training signals and slow convergence for the Discriminator, while a large m makes it harder for the Discriminator to effectively assess the quality of hyperedges. These results suggest that a balanced choice of m is crucial for effective interaction between the encoder and the Discriminator.

5 Conclusion

In this paper, we propose a *Critical Node*-aware hypergraph contrastive learning method (CNHCL), which is the first attempt to leverage hyperedge prediction to retain critical nodes in augmented hypergraphs. CNHCL first employs contrastive loss to update the encoder, further generating hyperedge embeddings to train the *Generator* and the *Hyperedge Prediction Discriminator*. Then, CNHCL utilizes the *Hyperedge Prediction Discriminator* to score all hyperedge embeddings, in order to update the candidate critical nodes and identified critical nodes. Afterwards, these critical nodes are retained in the augmented hypergraphs, naturally guiding better hypergraph representations for downstream tasks. Extensive experiments on various datasets validate that our model guides better hypergraph representations for downstream tasks and achieves significant superiority against state-of-the-art methods.

Contribution Statement

Zhuo Li and Yuena Lin contribute equally to this work. Gengyu Lyu is the corresponding author.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62306020), the Young Elite Scientist Sponsorship Program by BAST (No. BYESS2024199), the CCF-NSFOCUS'Kunpeng'Research Fund (No. CCF-NSFOCUS 2024013), the Beijing Natural Science Foundation (No. L244009), the National Key Research and Development Program of China (No. 2023YFB3107100), and the Project of Civi1 Aviation Intelligent Service Platform (No. P4CM02).

References

[Bai *et al.*, 2021] Song Bai, Feihu Zhang, and Philip H.S. Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021.

[Chien *et al.*, 2022] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. In *International Conference on Learning Representations*, pages 1–24, 2022.

[Feng *et al.*, 2019] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *AAAI conference on artificial intelligence*, pages 3558–3565, 2019.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

[Han *et al.*, 2024] Jiadi Han, Yufei Tang, Qian Tao, Yuhan Xia, and Liming Zhang. Dual homogeneity hypergraph motifs with cross-view contrastive learning for multiple social recommendations. *ACM Transactions on Knowledge Discovery from Data*, 18(6):1–24, 2024.

[Huang and Yang, 2021] Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. In *International Joint Conference on Artificial Intelligence*, pages 2563–2569, 2021.

[Hwang *et al.*, 2022] Hyunjin Hwang, Seungwoo Lee, Chanyoung Park, and Kijung Shin. Ahp: Learning to negative sample for hyperedge prediction. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2237–2242, 2022.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, pages 1–14, 2017.

[Lee and Chae, 2024] Jongsoo Lee and Dong-Kyu Chae. Multi-view mixed attention for contrastive learning on hypergraphs. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2543–2547, 2024.

[Lee and Shin, 2023] Dongjin Lee and Kijung Shin. I'm me, we're us, and i'm us: Tri-directional contrastive learning on hypergraphs. In *AAAI Conference on Artificial Intelligence*, pages 8456–8464, 2023.

[Lee *et al.*, 2024] Geon Lee, Soo Yong Lee, and Kijung Shin. Villain: Self-supervised learning on homogeneous hypergraphs without features via virtual label propagation. In *ACM on Web Conference*, pages 594–605, 2024.

[Lin *et al.*, 2021] Zhuoyi Lin, Lei Feng, Rui Yin, Chi Xu, and Chee Keong Kwoh. Glimg: Global and local item graphs for top-n recommender systems. *Information Sciences*, 580:1–14, 2021.

[Qian *et al.*, 2024] Yiyue Qian, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. Dual-level hypergraph contrastive learning with adaptive temperature enhancement. In *ACM Web Conference*, page 859–862, 2024.

[Ren *et al.*, 2024] Guojing Ren, Xiao Ding, Xiao-Ke Xu, and Hai-Feng Zhang. Link prediction in multilayer networks via cross-network embedding. In *AAAI Conference on Artificial Intelligence*, pages 8939–8947, 2024.

[Saifuddin *et al.*, 2023] Khaled Mohammed Saifuddin, Briana Bumgardner, Farhan Tanvir, and Esra Akbas. Hygnn: Drug-drug interaction prediction via hypergraph neural network. In *International Conference on Data Engineering*, pages 1503–1516, 2023.

[Veličković *et al.*, 2019] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, pages 1–17, 2019.

[Wan *et al.*, 2023] Sheng Wan, Yibing Zhan, Shuo Chen, Shirui Pan, Jian Yang, Dacheng Tao, and Chen Gong. Boosting graph contrastive learning via adaptive sampling. *IEEE Transactions on Neural Networks and Learning Systems*, 35(11):1–13, 2023.

[Wang *et al.*, 2023] Chenxu Wang, Xin Wang, Zhao Li, Zirui Chen, and Jianxin Li. Hyconve: A novel embedding model for knowledge hypergraph link prediction with convolutional neural networks. In *ACM Web Conference*, pages 188–198, 2023.

[Wei *et al.*, 2022] Tong Wei, Hai Wang, Weiwei Tu, and Yufeng Li. Robust model selection for positive and unlabeled learning with constraints. *Science China Information Sciences*, 65(11):212101, 2022.

[Wu *et al.*, 2024] Hanrui Wu, Nuosi Li, Jia Zhang, Sentao Chen, Michael K Ng, and Jinyi Long. Collaborative contrastive learning for hypergraph node classification. *Pattern Recognition*, 146:109995, 2024.

[Xu *et al.*, 2023] Jie Xu, Shuo Chen, Yazhou Ren, Xiaoshuang Shi, Hengtao Shen, Gang Niu, and Xiaofeng Zhu. Self-weighted contrastive learning among multiple views for mitigating representation degeneration. In *Advances in Neural Information Processing Systems*, pages 1119–1131, 2023.

[Yadati *et al.*, 2019] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in Neural Information Processing Systems*, 32:850, 2019.

[Yadati *et al.*, 2020] Naganand Yadati, Vikram Nitin, Madhav Nimishakavi, Prateek Yadav, Anand Louis, and Partha Talukdar. Nhp: Neural hypergraph link prediction. In *ACM International Conference on Information & Knowledge Management*, page 1705–1714, 2020.

[You *et al.*, 2020] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Advances in Neural Information Processing Systems*, pages 5812–5823, 2020.

[Zhang *et al.*, 2020] Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. In *International Conference on Learning Representations*, pages 1–7, 2020.

[Zhu *et al.*, 2020] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. In *ICML Workshop on Graph Representation Learning and Beyond*, pages 1–17, 2020.

[Zhu *et al.*, 2023] Jianian Zhu, Weixin Zeng, Junfeng Zhang, Jiuyang Tang, and Xiang Zhao. Cross-view graph contrastive learning with hypergraph. *Information Fusion*, 99:101867, 2023.

[Zhu *et al.*, 2024] Yonghua Zhu, Lei Feng, Zhenyun Deng, Yang Chen, Robert Amor, and Michael Witbrock. Robust node classification on graph data with graph and label noise. In *AAAI conference on artificial intelligence*, pages 17220–17227, 2024.