Hypergraph Structure Learning for Hypergraph Neural Networks

Derun Cai\textsuperscript{1,2}, Moxian Song\textsuperscript{1,2}, Chenxi Sun\textsuperscript{1,2}, Baofeng Zhang\textsuperscript{1,2}, Shenda Hong\textsuperscript{3,4,*} and Hongyan Li\textsuperscript{1,2,*}

\textsuperscript{1}Key Laboratory of Machine Perception (Ministry of Education), Peking University, Beijing, China.
\textsuperscript{2}School of Electronics Engineering and Computer Science, Peking University, Beijing, China.
\textsuperscript{3}National Institute of Health Data Science, Peking University, Beijing, China.
\textsuperscript{4}Institute of Medical Technology, Health Science Center of Peking University, Beijing, China.
\{cdr,boffinzhang\}@stu.pku.edu.cn, \{songmoxian,sun,chenxi,hongshenda,leehy\}@pku.edu.cn

Abstract

Hypergraphs are natural and expressive modeling tools to encode high-order relationships among entities. Several variations of Hypergraph Neural Networks (HGNNs) are proposed to learn the node representations and complex relationships in the hypergraphs. Most current approaches assume that the input hypergraph structure accurately depicts the relations in the hypergraphs. However, the input hypergraph structure inevitably contains noise, task-irrelevant information, or false-negative connections. Treating the input hypergraph structure as ground-truth information unavoidably leads to suboptimal performance. In this paper, we propose a Hypergraph Structure Learning (HSL) framework, which optimizes the hypergraph structure and the HGNNs simultaneously in an end-to-end way. HSL learns an informative and concise hypergraph structure that is optimized for downstream tasks. To efficiently learn the hypergraph structure, HSL adopts a two-stage sampling process: hyperedge sampling for pruning redundant hyperedges and incident node sampling for pruning irrelevant incident nodes and discovering potential implicit connections. The consistency between the optimized structure and the original structure is maintained by the intra-hyperedge contrastive learning module. The sampling processes are jointly optimized with HGNNs towards the objective of the downstream tasks. Experiments conducted on 7 datasets show that HSL outperforms the state-of-the-art baselines while adaptively sparsifying hypergraph structures.

1 Introduction

Graphs are ubiquitous across the real world and Graph Neural Networks (GNNs) [Kipf and Welling, 2017; Hamilton et al., 2017; Velickovic et al., 2018] have shown superiority in modeling the graph structure and node features. Since graphs only capture the pairwise relationship among entities, it is desirable to capture high-order interactions to learn effective node representations. The hypergraph, where each hyperedge can connect any number of nodes, can encode high-order relationships and has drawn increasing attention from many fields such as recommendation systems [Yu et al., 2021] and computer vision [Gao et al., 2020]. A series of Hypergraph Neural Networks (HGNNs) [Chien et al., 2021; Huang and Yang, 2021; Feng et al., 2019], which generalize GNNs to hypergraphs, has been proposed based on different message passing schemes.

Most HGNNs follow a two-stage message passing scheme, i.e., iteratively update hyperedge representations by aggregating nodes’ information and then update the node representations by aggregating incident hyperedges’ information. The message passing process of these methods relies on the input hypergraph structures. Hence, the hypergraph structure is crucial for HGNNs to learn expressive node representations. Most existing HGNNs simply presume that the hypergraph structure is perfect, which is not always true. Specifically, the quality of hypergraph structures needs to be improved in the following 3 aspects.

• Firstly, the hypergraph structures may contain task-irrelevant information, noise, or even mistakes, which are unnecessary information and need to be removed.
• Secondly, the hypergraph structures may contain false-negative connections, i.e., some nodes should be included in certain hyperedges but they are not. The implicit connections, which describe the underlying relationships and can be viewed as an enhancement of the original structure, are also important and need to be included in the hypergraph.
• Finally, the hypergraph structures may not be ideal for the downstream tasks since the raw structure only describes the relations among nodes from the input feature space. It may fail to reflect relations among nodes from high-level feature space. In addition, some connections in the hypergraph can be discarded while others need to be emphasized depending on the downstream tasks.

Therefore, it is important to jointly optimize the hypergraph structure and the HGNNs towards downstream tasks. The hyperedge and node representations learned by HGNNs provide information for learning better hypergraph structures. The newly learned hypergraph structures, in return, serve as the input of HGNNs to learn better hyperedge and node representations.

*Contact Author
Although hypergraph structure learning is important, it is challenging and unexplored. Many graph structure learning methods have been proposed to adaptively learn graph structures for GNNs. Unlike the graph structure learning problem where only pair-wise relationships are considered, we need to model high-order relationships in hypergraphs and update the structure where a hyperedge may connect any number of nodes. Due to the complexity and the inherent nature of hypergraphs, these methods cannot be directly applied to hypergraphs.

In this work, we present a Hypergraph Structure Learning (HSL) framework for HGNNs, aiming at jointly learning the hypergraph structure and the HGNNs that are optimized towards the downstream tasks. Specifically, we propose a two-stage sampling process for learning the hypergraph structures. In the first stage, the hypergraph structures are adjusted at a coarse-grained level via hyperedge sampling. In the second stage, the hypergraph structures are adjusted at a fine-grained level via incident node sampling. Potential implicit connections are discovered based on the hyperedge and node representations learned by HGNNs in this stage. The discrete sampling processes are differentiable with the help of the Gumbel Softmax trick. To maintain the consistency between the optimized hypergraph structure and the original input structure, intra-hyperedge contrastive learning is proposed to maximize the mutual information between the node representations from different hypergraph structures. These three modules, together with the backbone HGNNs, play a synergistic effect in learning better hypergraph structures and HGNNs. Experimental results on a variety of datasets demonstrate the effectiveness of our design. The contribution of this work can be summarized as follows:

- To the best of our knowledge, this is the first work to jointly optimizes the discrete hypergraph structure and HGNNs in an end-to-end way, such that our method has comparable or better performance and induces sparsified hypergraph structures.
- The proposed HSL, which consists of hyperedge sampling, incident node sampling, and intra-hyperedge contrastive learning, provides a framework for jointly optimizing the hypergraph structure and HGNNs.
- The extensive experiments conducted on 7 datasets show that HSL outperforms the state-of-the-art HGNNs while adaptively sparsifying hypergraph structures according to downstream tasks.

2 Related Work

2.1 Hypergraph Neural Networks

Graphs have limitations for representing high-order relationships. In a hypergraph, the complex relationships are encoded by hyperedges that can connect any number of nodes. [Zhou et al., 2006] introduced hypergraph to model high-order relations for semi-supervised classification and clustering of nodes. Recent works [Feng et al., 2019; Yadati et al., 2019; Bai et al., 2021] have designed different message passing strategies and have shown superiority in higher-order representation learning for graphs. [Feng et al., 2019] proposes the first HGNN method, which uses the clique expansion to transform hypergraphs to graphs and applies graph embedding techniques. UniGNN [Huang and Yang, 2021] proposes to unify hypergraph and GNN models using hypergraph star expansion. Many variations of GNNs can be incorporated in UniGNN. [Chien et al., 2021] proposes a general HGNN framework that implements HGNN layers as compositions of two multiset functions and covers propagation methods of most existing HGNNs.

2.2 Graph Structure Learning

Many graph structure learning methods have been proposed to learn ideal graph structures for GNNs [Zhu et al., 2021; Franceschi et al., 2019; Chen et al., 2020b; Luo et al., 2021; Zheng et al., 2020]. Some of these methods directly optimized the adjacency matrix with certain constraints such as low-rank, sparsity, and smoothness. Other methods model the existence of the edge as a metric function between pairwise nodes or a certain distribution.

Hypergraph structure learning is relatively under-explored. DHSL [Zhang et al., 2018] simultaneously optimizes the label projection matrix and the hypergraph structure iteratively, but it neither scale to large hypergraphs nor utilizes high-level representations. DHGNN [Jiang et al., 2019] uses K-Means and KNN to adaptively construct hypergraph, which is not optimized along with the HGNNs in an end-to-end way. HERALD [Zhang et al., 2021] optimize hypergraph Laplacian matrix instead of the discrete hypergraph incidence matrix, which means it can not be easily combined with the general HGNNs framework based on two-stage message passing.

3 Method

As illustrated in Figure 1, we present Hypergraph Structure Learning (HSL) framework for HGNNs. Inspired by the inherent properties of hypergraphs, the hypergraph structure learning process can be viewed as a two-stage sampling process: hyperedge sampling and incident node sampling. Leveraging the idea that the node representations learned from the original hypergraph and optimized hypergraph should be consistent, intra-hyperedge contrastive learning is applied.

3.1 Problem Definition

In this paper, we focus on the node classification task on hypergraphs, which map each node to a set of labels. Let \( G = (V, \mathcal{E}) \) denote the hypergraph with node set \( V \) and hyperedge set \( \mathcal{E} \). Each hyperedge \( e \in \mathcal{E} \) is a subset of \( V \). Let \( N = |V| \) denotes the number of nodes and \( E = |\mathcal{E}| \) denotes the number of hyperedges. \( N_i = \{e \in \mathcal{E} | i \in e\} \) denotes the incident hyperedges of node \( i \). The incident matrix represents the hypergraph structure and is denoted as \( H \in \{0, 1\}^{N \times E} \), where \( H_{ij} = 1 \) indicates that hyperedge \( j \) contains node \( i \). The input node features are denoted as \( X \in \mathbb{R}^{N \times d_x} \), where \( d_x \) is the dimension of node features. The goal of node classification on hypergraphs can be formulated as learning an HGNN \( F(X, \hat{H}) \) mapping hypergraph structure and node features to node labels. The goal of hypergraph structure learning is to learn the HGNN \( F(X, \hat{H}) \) and optimal hypergraph structure \( \hat{H} \) simultaneously.
3.2 HGNNs

HGNNs can be viewed as a two-stage message passing procedure [Huang and Yang, 2021; Chien et al., 2021], which can be formulated as follows.

\[
\begin{align*}
\mathbf{h}_j &= f_{\mathcal{V} \rightarrow \mathcal{E}}(\{\mathbf{x}_k\}_{k \in e_j}) \\
\tilde{\mathbf{x}}_i &= f_{\mathcal{E} \rightarrow \mathcal{V}}(\{\mathbf{h}_k\}_{k \in \mathcal{N}_i}, \mathbf{x}_i)
\end{align*}
\]

\(f_{\mathcal{V} \rightarrow \mathcal{E}}\) and \(f_{\mathcal{E} \rightarrow \mathcal{V}}\) are two permutation-invariant functions which aggregating information from nodes and hyperedges respectively. In the first stage of message passing, node representations \(\{\mathbf{x}_k\}_{k \in e_j}\) within each hyperedge are aggregated by \(f_{\mathcal{V} \rightarrow \mathcal{E}}\) to form hyperedge representation \(\mathbf{h}_j\). In the second stage, the node representations \(\tilde{\mathbf{x}}_i\) are updated with its incident hyperedges \(\{\mathbf{h}_k\}_{k \in \mathcal{N}_i}\) using \(f_{\mathcal{E} \rightarrow \mathcal{V}}\).

Recent works on HGNN have proposed many implementations for \(f_{\mathcal{V} \rightarrow \mathcal{E}}\) and \(f_{\mathcal{E} \rightarrow \mathcal{V}}\). HSL adopts AllSetTransformer [Chien et al., 2021] as the backbone network since it is a general and expressive framework of HGNNs.

3.3 Hyperedge Sampling

Inspired by the nature of hypergraph, HSL firstly performs hyperedge sampling to filter out task-irrelevant hyperedges. As a hyperedge can connect any number of nodes, hypergraph structure learning is performed in a coarsen-grained level in this stage. To sample hyperedges, we introduce a binary mask vector \(\mathbf{m} \in \{0, 1\}^E\), with \(\mathbf{m}_i\) denotes the existence of the hyperedge \(i\). Each binary number \(\mathbf{m}_i\) is considered to be drawn from a Bernoulli distribution, parameterized by \(\mathbf{z}_i\), i.e., \(\mathbf{m}_i \sim Bern(\mathbf{z}_i)\). \(\mathbf{z}^e \in [0, 1]^E\), the Bernoulli distribution parameter vector for hyperedges, is trainable parameters and is optimized jointly with the downstream task. Since it describes the importance of each hyperedge, a small value of \(\mathbf{z}_i\) indicates that the hyperedge \(i\) is likely to be noise and should be removed.

The discrete sampling process is not differentiable. To support a learnable and discrete sampling process, HSL applied the Gumbel Softmax trick [Maddison et al., 2017; Jang et al., 2017] for gradient approximation. For each hyperedge \(i\), the Gumbel Softmax is applied to generate the differentiable binary mask \(\mathbf{m}_i\). \(\epsilon^0, \epsilon^1\) is Gumbel random noise sampled from \(Gumbel(0, 1)\) and \(\tau\) is the temperature of the softmax distribution. \(\sigma\) denotes the sigmoid function.

\[
\mathbf{m}_i = \sigma\left(\frac{1}{\tau}(\log(\frac{\mathbf{z}_i^e}{1 - \mathbf{z}_i^e}) + (\epsilon^0 - \epsilon^1))\right),
\]

\(\epsilon^0, \epsilon^1 \sim Gumbel(0, 1)\)

With the sampling mask \(\mathbf{m}\), the incident matrix \(\mathbf{H}\) of the hypergraph structure after hyperedge sampling can be formulated as follows. \(\Gamma : \mathbb{R}^E \rightarrow \mathbb{R}^{N \times E}\) denotes the broadcasting operator, which repeats the mask vector \(\mathbf{m}\) for \(N\) times to construct a binary mask matrix \(\mathbf{M}^e \in \mathbb{R}^{N \times E}\) for hyperedge sampling. \(\odot\) denotes the Hadamard product.

\[
\mathbf{H} = \mathbf{M}^e \odot \mathbf{H}, \quad \mathbf{M}^e = \Gamma(\mathbf{m})
\]

3.4 Incident Node Sampling

In the hyperedge sampling stage, unnecessary hyperedges are pruned. In this stage, hypergraph structure learning is performed at a fine-grained level. Incident node sampling aims at refining the hypergraph structure via removing irrelevant nodes from hyperedges and adding potential nodes to hyperedges.

Before sampling incident nodes, the input hypergraph structure can be enhanced if the implicit connections between nodes and hyperedges are discovered. The default fixed hypergraph structure is possibly noisy with missing false-negative connections. To learn the implicit connections, it is necessary to augment the \(\mathbf{H}\) with a residual hypergraph structure \(\Delta \mathbf{H}\). Based on the node representation \(\mathbf{x}\) and the hyperedge representation \(\mathbf{h}\) extracted by HGNNs, we can find the most similar node-hyperedge pairs, which are potential connections with a high possibility. Let \(\mathbf{S} \in \mathbb{R}^{N \times E}\) denote the similarity matrix between nodes and hyperedges. To increase the expressive power, HSL adopted the multi-head weighted cosine similarity as the metric function. The similarity is
computed as the average of \( n_c \) independent similarity scores using \( n_c \) weight vectors \( \{ w_i \}_{i=1}^{n_c} \).

\[
S_{ij} = \frac{1}{n_c} \sum_{i=1}^{n_c} \cos(w_i \odot \hat{x}_{ij}, w_i \odot h_j) \quad (4)
\]

To reduce the expensive computation and noise, the residual hypergraph structure \( \Delta H \) is sparsified via a threshold \( p_{add} \), which denotes the proportion of the connections (the number of non-zero values in the incident matrix) between the residual hypergraph structure \( \Delta H \) and original structure \( H \). Specifically, the connections between nodes and hyperedges are selected according to the largest values in the similarity matrix \( S \) at the ratio \( p_{add} \). Since the residual hypergraph structure \( \Delta H \) aims at discovering implicit connections using learned node and hyperedge representations, the original connections in the input hypergraph structure are not included in \( \Delta H \). Let \( \text{top}(S, p_{add}) \) denote the function which select the largest items in the matrix \( S \) according to the ratio \( p_{add} \). The computation of \( \Delta H \) can be formulated as follows.

\[
\Delta H_{ij} = \begin{cases} 1, & H_{ij} = 0 \text{ and } S_{ij} \in \text{top}(S, p_{add}), \\ 0, & H_{ij} = 1 \\ \end{cases} \quad (5)
\]

\( H + \Delta H \) represent the enhanced hypergraph structure, which includes the original connections and possible implicit connections. To further refine the hypergraph structure, HSL performs incident node sampling based on the enhanced hypergraph structure \( H + \Delta H \). Similar to hyperedge sampling, we introduce a binary mask matrix \( \hat{M}^e = \mathbb{1}_{N \times E} \) for incident node sampling. Each binary number of \( \hat{M}^e_{ij} \) is also considered to be drawn from a Bernoulli distribution, parameterized by \( Z_{ij}^e \). \( Z^e \in [0, 1]^{N \times E} \) denotes the Bernoulli distribution parameter matrix. A small value of \( Z_{ij}^e \) indicates that the connections between hyperedge \( j \) and node \( i \) are more likely to be unnecessary and node \( i \) should be removed from hyperedge \( j \). \( Z^e \) is computed by an MLP taking node and hyperedge representations as inputs. In other words, the semantic information from node and hyperedge representation is utilized to decide the existence probability of the connections between nodes and hyperedges.

\[
Z_{ij}^e = \sigma(\text{MLP}([\hat{x}_i] || h_j)) \quad (6)
\]

To support the differentiable Bernoulli sampling process, the Gumbel Softmax trick is applied for binary mask matrix \( M^e \), which can be obtained via

\[
M_{ij}^e = \sigma\left(\frac{1}{\tau} \left( \log\left( \frac{Z_{ij}^e}{1 - Z_{ij}^e} \right) + (e^0 - e^1) \right) \right),
\]

\( e^0, e^1 \sim \text{Gumbel}(0, 1) \).

After performing incident node sampling with mask matrix \( M^e \), the final hypergraph structure \( \hat{H} \) can be obtained via

\[
\hat{H} = M^e \odot M^r \odot (H + \Delta H), \quad M^r = \Gamma(m). \quad (8)
\]

A Previous study [Huang and Yang, 2021] points out that self-loops are crucial in HGNNs and the performance drop significantly for most datasets when the hypergraph is self-looped. In the data preprocessing stage, self-loops are added to the hypergraphs where each self-loop only contains one node. The rationale is that the self-loops provide a shortcut to perform message passing only using the representation of the target node in the previous layer. If the self-loop of a certain node is missing, the representation of that node may be easily misguided by neighborhood nodes. The model degenerates to MLP if the hypergraph only contains self-loops and every node is isolated. To enhance the expressive power of the model, the self-loops should not be included in both of the sampling processes. Since the self-loops are included in the hypergraph, the number of hyperedges \( E \) should be greater or equal to the number of nodes \( N \). Let the last \( N \) hyperedges \( \{ e_i \}_{i=N-E-N}^E \) denote the self-loops. \( I \in \{0, 1\}^{N \times E} \) denotes the incident matrix of all the self-loops. The final hypergraph structure \( \hat{H} \) can be formulated as follows.

\[
\hat{H} = M^e \odot M^r \odot (H + \Delta H) + I, \quad I_{ij} = \begin{cases} 1, & i + E = j + N, \\ 0, & i + E \neq j + N \end{cases} \quad (9)
\]

### 3.5 Intra-Hyperedge Contrastive Learning

To maintain the consistency between the original hypergraph structure \( H \) and the altered structure \( \hat{H} \), intra-hyperedge contrastive learning is proposed. On the one hand, the node representations extracted by HGNNs based on \( H \) and \( \hat{H} \) should maintain the same or similar semantic information. On the other hand, the message passing scheme may result in an over-smoothing issue [Li et al., 2018] and the representations of the node in the same hyperedge should be distinguishable from each other. The node representations extracted by HGNNs based on \( H \) and \( \hat{H} \) are different views of the same node and they should be consistent. Intra-hyperedge contrastive learning is applied to minimize the distance between the representations of the same node from different views and maximize the distance between the representations of different nodes in the same hyperedge.

Let \( \hat{x}, x \) denote the node embedding from hypergraph structure \( \hat{H}, H \) respectively. \( T_i = \{j \mid e_i \in N_j\} \) denote the set of neighbour nodes which are in the same incident hyperedges as node \( i \). HSL follows SimCLR [Chen et al., 2020a] and adopts the InfoNCE loss [Gutmann and Hyvärinen, 2010], which maximize the mutual information between the node representation from different views. Let \( \phi \) denotes the similarity function for contrastive learning, which is implemented as cosine similarity. The intra-hyperedge contrastive loss \( L_{CL} \) is formulated as follows.

\[
L_{CL}(x_i) = -\log \frac{\exp(\phi(x_i, \hat{x}_i))}{\sum_{j \in T_i, j \neq i} \exp(\phi(x_i, x_j)) + \sum_{j \in T_i} \exp(\phi(x_i, \hat{x}_j))} \quad (10)
\]

The hyperedge sampling and incident node sampling process are implemented by mask matrix \( M^e \) and \( M^r \) respectively. Let \( \lambda \) denotes the hyper-parameter balancing the cross-entropy loss function \( L_T \) and the intra-hyperedge contrastive loss \( L_{CL} \). The sampling processes are jointly optimized with the loss function of the downstream task \( L_T \) during training.
The final training objective $L$ can be formulated as follows.

$$L = \frac{1}{N} \sum_{i=1}^{N} (L_T(F(x_i, H), y_i) + \lambda L_{CL}(x_i))$$ \hspace{1cm} (11)

where $H = M^r \odot M^v \odot (H + \Delta H) + I$

### 4 Experiments

#### 4.1 Experimental Setups

To evaluate the effectiveness of HSL, experiments are conducted on 7 public datasets: Cora, Citeseer, Coauthor-Cora, Coauthor-DBLP [Yadati et al., 2019; Rossi and Ahmed, 2015; Sen et al., 2008], NTU2012 [Chen et al., 2003], 20Newsgroups [Zhou et al., 2006]. The statistics of the datasets are shown in Table 2. In Citeseer and Cora dataset, the co-authorship hypergraphs are constructed where a hyperedge connects all documents co-authored by an author. In Coauthor-Cora and Coauthor-DBLP dataset, the co-citation hypergraphs are constructed where a hyperedge connects all documents cited by an author.

HSL is compared against 6 baseline models: (1) MLP: Only using the node feature as input and ignoring the hypergraph structure, (2) DHGNN [Jiang et al., 2019]: Dynamic Hypergraph Neural Networks, a method using KNN and K-Means to adaptively update hypergraph structure, (3) CEGAT: Using clique-expansion to transform hypergraphs to graphs and applying GAT [Velickovic et al., 2018], (4) HGNN [Feng et al., 2019]: Hypergraph Neural Networks, (5) UniGCNII [Huang and Yang, 2021], (6) AllSetTransformer [Chien et al., 2021].

We perform semi-supervised node classification tasks in the transductive setting on these datasets. In the semi-supervised node classification task, the labels for the test nodes are predicted by HGNNs based on the hypergraph structure, the input features of all nodes, and limited node labels for training. The datasets are randomly split into the training, validation, and test set in a 0.5:0.25:0.25 ratio. The experiments are repeated with 20 random data splits. The number of HGNN layers is set to 1. We use the Adam optimizer with a learning rate tuned over $\{0.001, 0.0001\}$ on the validation set. The hidden dimension of HGNNs on different datasets is tuned over $\{64, 128, 256, 512\}$. The hyperparameter $p_{odd}$ for enhancing hypergraph structure is tuned over $\{0, 0.01, 0.02, 0.05\}$. All programs are implemented using the Pytorch Geometric library (PyG) [Fey and Lenssen, 2019] with PyTorch 1.8 on an Nvidia RTX 3090 GPU.

### 4.2 Experimental Results

#### The Accuracy of the Node Classification tasks.

The performance of all methods is summarized in Table 1. The experimental results show that HSL has the best overall performance when compared to the state-of-the-art. With the structure learning modules, HSL effectively optimizes the hypergraph structure. The HGNNs can learn better node representations from better hypergraph structure, leading to better performance on the downstream node classification tasks. The expressive ability of methods that use fixed hypergraph structure as input (CEGAT, HGNN, UniGCNII, AllSetTransformer) is limited by the hypergraph structure which is not suitable for downstream tasks. DHGNN [Jiang et al., 2019] applied KNN and K-Means to dynamically adjust the hypergraph structure. It achieves promising accuracy on Cora and Citeseer dataset, but its performance is not desirable on other datasets. DHGNN is not optimized in an end-to-end way and the degree of the hyperedges or nodes is fixed according to a pre-defined hyper-parameter $k$, which greatly hinders the representation ability of the model. Jointly optimized with the downstream task, our proposed HSL can dynamically adjust the hypergraph structure with no constraint on the degrees or neighbors of the hyperedges and nodes.

#### The Pruning Rate of the HSL.

To further examine the effect of the HSL, we compute the pruning rate $r^v$ of the learned hypergraph structure $H$. $r^v$ denote the sparsity of the optimized hypergraph structure $H$ compared with the original hypergraph structure $H$, i.e., the ratio between the number of non-zero values in $H$ and $H$. As shown in Table 1, HSL can effectively sparsify the input hypergraph structure while achieving the best performance at the same time, indicating that HSL can keep the most task-relevant and informative part of the hypergraph structure. HSL achieves the
average pruning rate of 59.7% and 11.31% for the Citeseer dataset. Noticed that the gap in the performance between the MLP and other hypergraph-based methods is narrow (smaller than 1%) for some datasets such as Citeseer, 20Newsgroups. It indicates that the hypergraph structures are not helpful to these HGNNs on the downstream task. Taking the fixed input hypergraph as ground truth may result in sub-optimal performance. HSL can dynamically adjust the hypergraph structure and prune the unnecessary hyperedges and connections, balancing the accuracy and the sparsity. In the Citeseer and 20Newsgroups datasets, the gap between MLP and hypergraph-based methods is the smallest. HSL achieves the highest pruning rate in those two datasets.

![Figure 2: The performance of HSL and AllSetTransformer on randomly corrupted Cora and 20Newsgroups datasets.](image)

**Robustness Against Noisy Datasets.** To evaluate the robustness of HSL to hypergraphs with noisy or missing connections, we construct hypergraphs where nodes are randomly added to or removed from hyperedges. We choose AllSetTransformer as the baseline model. Experiments are conducted on different levels of noise, ranging from -100% to 100%. -100% means removing all the connections between hyperedges and nodes and 100% means adding the same number of connections as the original hypergraph structure. As shown in Figure 2, HSL outperforms AllSetTransformer at a different level of noise.

![Figure 3: Ablation Study.](image)

**Ablation Study.** We analyze the influence of hyperedge sampling, incident node sampling, and intra-hyperedge contrastive learning modules of our proposed HSL. As shown in Figure 3, all of the three modules contribute to the performance. The following 3 variations of HSL are compared: (1) Base: The backbone HGNN without any hypergraph structure learning modules, (2)+edge: The backbone HGNN combined with hyperedge sampling module, (3)+node: The backbone HGNN combined with incident node sampling module, (4)-CL: HSL without the intra-hyperedge contrastive learning module. All of these modules are important. The hyperedge sampling module aims at learning the hypergraph structure at a coarse-grained level while the incident node sampling module aims at learning the hypergraph structure at a fine-grained level. The intra-hyperedge contrastive learning module plays an important role in maintaining the consistency between the optimized structure and the original structure. These three modules play a synergic effect to achieve the best performance.

![Figure 4: Visualization of the hypergraph structure optimized by HSL and the original hypergraph structure using star expansion. The blue dots denote the node while the red dots denote the hyperedges. The edge denotes the connection between hyperedges and notes.](image)

**Visualization.** We visualized the hypergraph structure optimized by HSL and the original hypergraph structure using star expansion. On the Citeseer dataset, HSL achieves a pruning rate of 11.31%. As shown in Figure 4, HSL significantly condenses the hypergraph structure while achieving better accuracy, indicating that HSL effectively prunes task-irrelevant connections and hyperedges.

![Visualization](image)

**5 Conclusion**

Recently, many variations of HGNNs have been proposed, generalizing GNNs to hypergraphs. HGNNs are capable of capturing higher-order relationships among entities and have achieved great success in many applications. However, the hypergraph structure on which HGNNs rely may not always be ideal. It may contain noise, task-irrelevant information, or false-negative connections. In this paper, we propose HSL, which consists of a hyperedge sampling module, incident node sampling module, and intra-hyperedge contrastive learning module, for jointly learning hypergraph structure and HGNNs. Unnecessary hyperedges and connections between hyperedges and nodes are pruned. In addition, hypergraph structures are enhanced by learning potential implicit connections. Extensive empirical studies demonstrate the superiority of our model.
References


