Preventing Over-Smoothing for Hypergraph Neural Networks

Guanzi Chen\textsuperscript{1*}, Jiyling Zhang\textsuperscript{2*}, Xi Xiao\textsuperscript{2}, Yang Li\textsuperscript{1}
\textsuperscript{1}Tsinghua-Berkeley Shenzhen Institute, Tsinghua University
\textsuperscript{2}Shenzhen International Graduate School, Tsinghua University
\{cgz20, zhangjiy20\}@mails.tsinghua.edu.cn, \{xiaox, yangli\}@sz.tsinghua.edu.cn

\textbf{Abstract}

In recent years, hypergraph learning has attracted great attention due to its capacity in representing complex and high-order relationships. However, current neural network approaches designed for hypergraphs are mostly shallow, thus limiting their ability to extract information from high-order neighbors. In this paper, we show both theoretically and empirically, that the performance of hypergraph neural networks does not improve as the number of layers increases, which is known as the over-smoothing problem. To avoid this issue, we develop a new deep hypergraph convolutional network called Deep-HGCN, which can maintain the heterogeneity of node representation in deep layers. Specifically, we prove that a $k$-layer Deep-HGCN simulates a polynomial filter of order $k$ with arbitrary coefficients, which can relieve the problem of over-smoothing. Experimental results on various datasets demonstrate the superior performance of the proposed model compared to the state-of-the-art hypergraph learning approaches.

\textbf{1 Introduction}

Graphs are ubiquitous and are widely used to represent interactions between entities. Graph Convolutional Networks (GCNs) generalizes convolutional neural networks to graph-structure data and have shown superior performance on many applications\cite{Kipf and Welling, 2017; Zhang et al., 2022c; Chen et al., 2022; Chen et al., 2021}. However, graphs can only represent pairwise associations, inadequate in analyzing high-order relationships in the real world. For example, co-authorship may involve more than two authors and communities in social networks often involve more than two persons. Naïvely squeezing such complex relationships into pairwise ones will inevitably lead to unexpected loss of information.

Hypergraph, a generalization of graph containing edges that can be incident to more than two vertices, serve as a natural tool to model such complex and high order relationships. For example, In a co-citation hypergraph, hypernodes represent papers and hyperedges mean the citation relationships. The ubiquity of complex relationships in the real world naturally encourages the study of hypergraph learning, including clustering of categorical data\cite{Zhou et al., 2007}, multi-label classification\cite{Sun et al., 2008}, image segmentation\cite{Kim et al., 2011}, image classification\cite{Yu et al., 2012}, mapping users across different social networks\cite{Tan et al., 2014}, and so on.

Furthermore, motivated by the fact that GCNs have been successfully applied to a wide range of applications\cite{Kipf and Welling, 2017; Chen et al., 2020}, many Hypergraph Convolutional Neural Networks (HGCNs) are proposed to obtain the representation of hypergraph\cite{Zhang et al., 2022b}. For example, HGNN\cite{Feng et al., 2019} has been proposed as the first deep learning method on hypergraph structure, employing hypergraph Laplacian to represent hypergraph from spectral perspective. \cite{Yadati et al., 2019} and \cite{Dong et al., 2020} proposed a new hypergraph Laplacian to construct HGCNs, respectively. Although they have achieved enormous success, most of the current HGCNs are shadow and the performance degrades when it goes deeper. Such a phenomenon also occurs in GCNs and this phenomenon is called over-smoothing\cite{Li et al., 2018}. To tackle this problem, many methods have been proposed, such as GCNII\cite{Chen et al., 2020}, Scattering GCN\cite{Min et al., 2020} and DGC\cite{Zhou et al., 2020}. However, to the best of our knowledge, few work has specially studied the over-smoothing problem on hypergraphs, i.e. whether it exists and if so, how to prevent it. The methods developed for GCNs can not be easily extended to hypergraphs, for GCNs are based on graph normalized Laplacians but not on hypergraph Laplacians.

In this paper, on one hand, we prove that the deep HGCNs will undergo over-smoothing from the two aspects: stable distribution and Dirichlet energy. Specifically, the vanilla HGCNN(i.e. HGNN\cite{Feng et al., 2019}) can be associated with a random walk and eventually converges to the stationary state. And the Dirichlet energy \cite{Cai and Wang, 2020} of hypernode embeddings will converge to zero, resulting in the loss of discriminative power. On the other hand, we develop a Deep Hypergraph Convolutional Network called Deep-HGCN to alleviate the problem of over-smoothing. Specifically, Deep-HGCN extends two simple but effective techniques: Initial residual and Identity mapping \cite{Chen et al.,...
to the hypergraph setting. Initial residual is designed to prevent the convergence of node state by improving the heterogeneity of node representation in deep layers and identity mapping is used to slow down the convergence of Dirichlet energy. At each layer, initial residual connects the input layer, while identity mapping adds an identity matrix to the parametric weight matrix. Furthermore, we prove that a k-layer Deep-HGCN model can express a polynomial spectral filter of order k with arbitrary coefficients, which prevents the random walk from converging to a stationary distribution.

The main contributions of this paper are summarized as follows:

1. We prove that HGNNs have over-smoothing issues from the perspective of random walk and Dirichlet energy. Extensive experimental results also verify this phenomenon.
2. We further propose a truly deep HGCNN called DeepHGCN, which adopts two simple yet effective techniques: Initial residual and Identity mapping. The fact that a k-layer Deep-HGCN model can express a polynomial spectral filter of order k with arbitrary coefficients proved Deep-HGCN overcomes the over-smoothing problem.
3. We have conducted experiments on various hypernode classification datasets. The experimental results indicated that Deep-HGCN consistently relieves the over-smoothing problem and outperforms state-of-the-art methods.

2 Related work

Hypergraph Learning Hypergraph learning is first introduced in a seminal work [Zhou et al., 2007], and has been used to encode high-order correlations in many applications, such as image retrieval [Huang et al., 2010], image segmentation [Kim et al., 2011], 3D model classification [Gao et al., 2020]. Inspired by the superior success of GCNs models, many hypergraph neural network are proposed to obtain the representation of hypergraphs [Li et al., 2022]. HGNN [Feng et al., 2019] first employ hypergraph Laplacian to represent hypergraph from spectral perspective. HyperGCN [Yadati et al., 2019] trains a GCN for semi-supervised learning on hypergraphs via a new hypergraph Laplacian, which converts a hypergraph to a simple graph. HNHN [Dong et al., 2020] designed a convolution network combined both hypernodes and hyperedges features, which based on the hypergraph normalization with cardinality. [Zhang et al., 2022a] propose a learnable hypergraph Laplacian module for updating hypergraph topology during training.

Oversmoothing in GNNs [Li et al., 2018] first identified the phenomenon that performance will instead degrade when the GCNs models go deeper as over-smoothing. And there are a few other works proposed to addressing oversmoothing. JKNet [Xu et al., 2018] proposed Jumping Knowledge Networks, jumping connections into final aggregation mechanism. APPNP [Klicpera et al., 2019] proposed a propagation scheme derived from personalized PageRank. DropEdge [Rong et al., 2019] proposed to randomly drop out certain edges from the input graph. PairNorm [Zhao and Akoglu, 2020] proposed a novel normalization layer. GCNII [Chen et al., 2020] extend vanilla GCN with initial residual and identity mapping. DGN [Zhou et al., 2020] presented a group normalization layer. ScatteringGCN [Min et al., 2020] proposed to augment conventional GCNs with geometric scattering transforms and residual convolutions.

3 Understanding Oversmoothing

In this part, we introduce the hypergraph Laplacian and the extend model HGNN [Feng et al., 2019]. Further, we simplified the HGNN to obtain a simple HGCCN called SHGCN via removing the nonlinear transition functions between each layer and only keep the final softmax function. Based on HGNN and SHGCN, we conclude that the HGCNNs suffer from over-smoothing problem.

Notations. Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, $\mathcal{V}$ is the vertex set in which each hypernode $i \in \mathcal{V}$ is associated with a feature vector $x_i \in \mathbb{R}^d$, where $X = [x_1, \ldots, x_{|\mathcal{V}|}]^T$ denotes the feature matrix, $\mathcal{E}$ is the hyperedge set, and $\mathcal{W}$ is the weight matrix of hyperedges.

The structure of hypergraph $\mathcal{G}$ can be denoted by an incidence matrix $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$ with each entry of $h(v, e)$, which equals 1 when $e$ is incident with $v$ and 0 otherwise. The degree of hypernode and hyperedge are defined as $d(v) = \sum_{e \in \mathcal{E}} h(v, e)$ and $\delta(e) = \sum_{v \in \mathcal{V}} h(v, e)$ which can be denoted by diagonal matrices $\mathbf{D}_v \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ and $\mathbf{D}_e \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$.

3.1 Preliminaries on Hypergraph Convolutions

Hypergraph Spectral Convolution [Zhou et al., 2007] introduce the normalized hypergraph laplacian matrix as $\Delta = I - \mathbf{P}$, where $\mathbf{P} = \mathbf{D}_e^{-\frac{1}{2}} \mathbf{H} \mathbf{W} \mathbf{D}_v^{-\frac{1}{2}}$. $\Delta$ is a symmetric positive semi-definite matrix with spectral decomposition $\Delta = \Phi \Lambda \Phi^T$. Then, the spectral convolution of a signal $x = (x_1, \ldots, x_n)$ and a filter $g$ can be denoted as $g * x = \Phi((\Phi^T g) \circ (\Phi^T x)) = \Phi g(\Lambda) \Phi^T x$. [Feng et al., 2019] indicate that the filter $g(\Lambda)$ can be approximated by a $K$-th order polynomial of $\Lambda$, and the hypergraph convolution can be further approximated by the $K$-th order polynomial of Laplacians

$$
\Phi g_{\theta} \Phi^T x \approx \Phi \left( \sum_{k=0}^{K} \theta_k \Lambda^k \right) \Phi^T x = \left( \sum_{k=0}^{K} \theta_k \Delta^k \right) x, \quad (1)
$$

where $\theta \in \mathbb{R}^{K+1}$. HGNN [Feng et al., 2019] sets $K = 1$, $\theta_0 = -\frac{1}{2} \theta$, $\theta_1 = \frac{1}{2} \theta \mathbf{P}$ to derive $g * x = \theta \mathbf{P} x$, and finally build a Hypergraph Convolutional Layer

$$
X^{(l+1)} = \sigma(\mathbf{P} X^{(l)} \Theta^{(l)}), \quad (2)
$$

where $\sigma$ denotes the non-linear activation function.

Simple Hypergraph Convolutional Network In order to better analyze the performance of the multi-layer hypergraph convolutional networks, referring to SGC [Wu et al., 2019], we remove the non-linear activation function between each layer in Eq. (2) and add the softmax in the final layer to obtain Simple HyperGraph Convolutional Network called SHGCN, of which each layer can be denoted as:

$$
X^{(l+1)} = \mathbf{P} X^{(l)} \Theta^{(l)} \quad (3)
$$
and the output can be formulated as:
\[ Y = \text{softmax}(P^T X^{(0)} \Theta), \]  
(4)

where \( \Theta = \prod_{l=0}^{L} \Theta^{(l)} \) is a learnable parameter matrix and \( X^{(0)} = X \), representing the input features. Actually a \( K \)-layer SHGCN simulates a polynomial filter of order \( K \) with fixed coefficients \( \theta = [\theta_0, ..., \theta_K] \) since \( P^Kx = (I - \Delta)^Kx = \left( \sum_{k=0}^{K} \theta_k \Delta^k \right)x \). The biggest difference between SHGCN and HGNN is that the removed activation function, which is usually taken ReLU and can not essentially affect the process of convolution. So the over-smoothing problem of HGNN can be explored by SHGCN directly. Then we will show that the fixed filter limits the ability of expressive power and lead to over-smoothing.

### 3.2 Studying Over-smoothing with Random Walk

According to [Zhou et al., 2007], each hypergraph can be associated with a natural random walk with transition matrix \( T = D_v^{-1} H W D_e^{-1} H^T \). The stationary distribution of the random walk is
\[ \pi = 1^T D_v / \text{vol} \mathcal{V} \]  
(5)

where \( 1 \in \mathbb{R}^{1 \times |\mathcal{V}|} \) is an all-one vector, \( \pi \in \mathbb{R}^{1 \times |\mathcal{V}|} \) is the probability vector, and \( \text{vol} \mathcal{V} \) is the sum of the degrees of the vertices in \( \mathcal{V} \). Then we can derive the stationary distribution w.r.t. \( P \).

**Lemma 1** If \( G = (\mathcal{V}, \mathcal{E}, \mathcal{W}) \) is a hypergraph, then the stationary distribution of a random walk with transition matrix \( P = D_v^{-1} H W D_e^{-1} H^T D_v^{-1} \) on \( G \) is
\[ \tilde{\pi} = 1^T D_v^\frac{1}{2} / \text{vol} \mathcal{V} \]  
(6)

The proof of Lemma 1 can be found in supplementary materials. Next, we build the connection between random walk and Laplacian smoothing to further explore the hypergraph convolution. Considering a one-layer SHGCN. It actually contains two steps.

1) Generating a new feature matrix \( Y \) from \( X \) by applying the hypergraph convolution:
\[ Y = PX \]  
(7)

2) Feeding the new feature matrix \( Y \) to a fully connected layer.

The hypergraph convolution Eq.(7) actually aggregates the message of neighbor hypernodes and reduces the heterogeneity of hypernodes representation, which can be regraded as a special form of Laplacian smoothing [Li et al., 2018]. Then we show that the Laplacian smoothing lead to the problem of over-smoothing. Let \( x_{*,j} \in \mathbb{R}^{|\mathcal{V}|} \) denote the \( j \)-th column of \( X \) base on lemma 1, We derive the following theorem.

**Theorem 1** Repeatedly applying Laplacian smoothing on \( x_{*,j} \), we have
\[ \lim_{l \to \infty} P^l x_{*,j} = \pi_j^T \]  
(8)

where \( \pi_j = \left( \sum_{i=1}^{|\mathcal{V}|} x_{i,j} \right) \tilde{\pi} \in \mathbb{R}^{1 \times |\mathcal{V}|} \).

The proof of Theorem 1 is provided in supplementary materials. The Theorem 1 means that the repeatedly applied Laplacian smoothing on \( x_{*,j} \) will converge to a stationary point, which only depends on the sum of initial features \( \sum_{i=1}^{|\mathcal{V}|} x_{i,j} \) and the stationary distribution \( \tilde{\pi} \) that is only related to hypergraph structure(i.e.,degree). That is to say, the \( l \)-th representation \( P^l X \) of SHGCN converge to the matrix \( \left[ \sum_{i=1}^{|\mathcal{V}|} x_{1,i} \tilde{\pi}^T , ..., \sum_{i=1}^{|\mathcal{V}|} x_{id} \tilde{\pi}^T \right] \in \mathbb{R}^{|\mathcal{V}| \times d} \) which only contains the information of initial features and degree of each nodes leads to over-smoothing. what’s worse, the information of initial features just the sum of column of the \( X \), which does not contain any hypernode feature. Such convergence suggests that the \( K \)-order polynomial filter \( P^K X \) with fixed coefficients cause the problem of over-smoothing.

### 3.3 Studying Over-smoothing with Dirichlet energy

[Cai and Wang, 2020] suggests that expressiveness of GCNs can be measured by the Dirichlet energy and the over-smoothing problem is cause by the fact that the Dirichlet energy of embeddings converges to zero. So in this part, We
will investigate the expressive power of HGCNNs via their Dirichlet Energy as the layer size tends to infinity.

**Definition 1** Dirichlet energy $E(f)$ of a scalar function $f \in \mathbb{R}^{d \times 1}$ on the hypergraph $G^l = (V, E, W)$ is defined as:

$$E(f) = f^T \Delta f = \frac{1}{2} \sum_{e \in E} \sum_{(u, v) \in e} w(e) h(u, e) h(v, e) \delta(e)$$

where $h(u, e) = \Delta(u) = \sum_{v \in V} \lambda_{uv}$ is the degree at vertex $e$.

For the features matrix $X = [x_1, ..., x_\nu]^T$, the Dirichlet energy is defined as $E(X) = tr(X^T \Delta X)$. Recall Eq.(2), the representation of $l$-th layer in HGN [Feng et al., 2019] can be denoted as:

$$X^{(l)} = \sigma(\cdots \sigma(PX^{(0)} \Theta^{(1)}) \cdots \Theta^{(l-1)})$$

Next we study the influence of $P$, $\Theta_l$ and the activation function $\sigma$ on the Dirichlet energy respectively.

**Lemma 2** The hypergraph energy is same as Definition 1, we have

1) $E(PX) \leq (1 - \lambda)^2 E(X)$, where $\lambda$ is the minimum non-zero eigenvalue of $\Delta$.

2) $E(X\Theta) \leq \|\Theta\|^2 \|E(X)\|$, where $\cdot$ represents the maximum singular value of the matrix.

3) $E(\sigma(X)) \leq E(X)$, where $\sigma$ is ReLU or Leaky-ReLU activation function.

The above Lemma 2 can be directly proved by the definition of hypergraph energy. More detail about the proof can be see in the supplementary materials. Thus, we denote $s_1 := \prod_{k=0}^{l-1} \|\Theta^{(k)}\|^2_2$ and $\tilde{\lambda} := (1 - \lambda)^2$ and deduce the main theorem as follow.

**Theorem 2** For any $l \geq 1$, we have $E(X^{(l)}) \leq s_l \tilde{\lambda} E(X^{(l-1)})$.

Based on Eq.(10) and Lemma 2, the theorem can be proved. The detail of proof can be found in the supplementary materials.

**Corollary 2.1** Let $s := \max_{1 \leq k \leq l} s_k$, we have $E(X^{(l)}) \leq s \tilde{\lambda} E(X)$. In particular, $E(X^{(l)})$ exponentially converges to zero when $s \tilde{\lambda} < 1$.

From the Corollary 2.1, we know the Dirichlet energy of the embedding of HGN will converge to zero if $s < 1/\tilde{\lambda}$, which means that the HGCNNs will suffer from over-smoothing. And the result also reveals that $s$ is a key point that we can operate to improve the expressive power of a multi-layer model.

## 4 Deep-HGCN: Tackling Over-Smoothing

Recall to section 3, we get two main reasons that cause the problem of hypergraph convolution: 1) The fixed coefficient filter; 2) The diminishing Dirichlet energy. So in the part, we mainly focus on these two problems to design the multi-layer model.

### 4.1 Deep-HGCN Model

In order to improve the expressive power of deep HGCNNs, we can make the fixed coefficients flexible, which is a key point to prevent the over-smoothing [Chen et al., 2020]. Here, we adopts two techniques: Initial residual and Identity mapping to obtain the truly deep network Deep-HGCN. The two techniques are proved effective in preventing over-smoothing in GNNs [Chen et al., 2020]. Thus, the $l$-th layer of Deep-HGCN can be denote as

$$X^{(l+1)} = \sigma((1 - \alpha_l)PX^{(l)} + \alpha_l X^{(0)}) \Theta^{(l)}_l,$$  \hspace{1cm} (11)

where $\alpha_l$ and $\beta_l$ are two hyper-parameters, $\Theta^{(l)}_l = (1 - \beta_l)I_l + \beta_l \Theta^{(l)}_l \in \mathbb{R}^{c \times c}$ is the learnable weight matrix with identity mapping. As for $X^{(0)}$, we can set to the linear transform of input feature $X \in \mathbb{R}^{\nu \times d}$ if $d$ is large ($h \leq d$). Recall that $P = D^{-\frac{1}{2}}_{v} HWD^{-\frac{1}{2}}_{e} H^T D^{-\frac{1}{2}}_{v}$ is the hypergraph convolution matrix. Comparing Eq.(2) with Eq.(11), we notice two differences: 1) Eq.(11) add a identity matrix $I_c$ to the weight $\Theta$; 2) Eq.(11) add the first layer representation $X^{(0)}$ to the current layer Laplacian smoothing representation $PX^{(l)}$.

**Initial Residual connection** [Bai et al., 2019] proposes the Skip Connection to connect the smoothed feature $PX^{(l)}$ and $X^{(0)}$. However, they also show the performance deterioration as the model become deeper but do not give the reason. We start from the perspective of the stationary distribution of random walk to reconsider the Skip Connection. Based on the analysis over-smoothing in section 3.2, we know over-smoothing washes away the signal from all the features, making them indistinguishable. So we add the representation of the first layer to each layer for compensating the heterogeneity of the hypernodes. The hyper-parameters $\alpha_l$ indicates how much the initial features information that each layer can carry. Despite we stack many layers, it can receive at least one layer of the model.

**Identity mapping** It seems that we can alleviate the over-smoothing by adding the initial residuals in each layers, however, in GNNs, APPNP [Klicpera et al., 2019] adopt the Initial Residual connection only obtain a shallow model, which still undergo the over-smoothing problem. And our experiments also show that the same is true for HGCNNs(Figure 3). In order to improve the performance of the Deep-HGCN...
when the number of layers increases, we add the identity matrix to weight matrix. The motivation as follow:

- **Slow down the Dirichlet energy convergence to zero.** According to the analysis of Dirichlet energy in section 3.3, we know that the rate of convergence of energy depends on the $s^K$, where $s = \sup_{1 \leq k \leq K} s_l$. Replacing the $\Theta^{(l)}$ by $\Theta^{(l)}$ would make the largest singular value of the $\Theta^{(l)}$ (i.e. $s_l$) close to 1 if we apply regularization on $\Theta^{(l)}$ and force the norm of it to small. Consequently, The trend of $s^K$ tending to zero will slow down, so that the model can alleviate the energy reduction.

- **Useful in designing deep network.** ResNet [He et al., 2016] adds the identity mapping to ensure the training error of a deeper model no more than its shallower counter-part. So similar to the motivation of it, we expect to achieve that the performance of deeper model at least same to the shallow model.

- **Effective in solving over-smoothing problems in GCNs.** GCNII [Chen et al., 2020] add the identity matrix to weight matrix and indicate the technique is effective in the semi-supervised task. Notice the hypergraph can be seen as the generalization of graph, we apply identity mapping to Deep-HGCN and expect it can work on hypergraph convolution.

To ensure the retard the Dirichlet energy deteriorate, we should set the $\beta_l$ decreases adaptively as the number of layers increases. Here, we adopt the setting $\beta_l = \log(\frac{1}{l} + 1)$ as suggested in [Chen et al., 2020].

### 4.2 Spectral Analysis

Recall the analysis in section 3.2, a $K$-layer SHGCN simulates a polynomial filter $(\sum_{l=0}^{K} \theta_l \Delta^l) x$ of order $K$ with fixed coefficients $\theta$ on the hypergraph spectral domain of $G$, and the fixed coefficients limits the ability of learning the distinguishable hypernode representations of a multi-layer HGCNNs and thus cause over-smoothing. Now, we prove that a $K$-layer Deep-HGCN simulates a polynomial of order $K$ with arbitrary coefficients $\theta$.

**Theorem 3** Suppose $x$ is a graph signal in the $G$, then a $K$-layer Deep-HGCN can express a $K$ order polynomial filter $(\sum_{l=0}^{K} \theta_l \Delta^l) x$ with arbitrary coefficients $\theta$.  

| Dataset Method       | 2 | 4 | 8 | 16 | 32 | 64 |
|----------------------|---|---|---|----|----|----|
| MLP                  | 50.63 | 51.03 | 50.42 | **51.34** | 50.74 | 50.81 |
| Cora HyperGCN        | **52.82** | 50.71 | 23.84 | 23.25 | 23.13 | 23.08 |
| (co-a) HGNN           | 66.36 | 59.47 | 25.69 | 26.27 | 25.26 | 27.34 |
| Deep-HGCN            | 73.53 | 74.08 | 74.52 | 74.50 | 74.91 | 75.08 |
| MLP                  | 77.34 | 77.31 | **77.41** | 77.32 | 77.34 | OOM |
| DBLP HyperGCN        | **70.11** | 58.72 | 22.87 | 21.73 | 22.18 | OOM |
| (co-a) HGNN           | **88.22** | 85.30 | 26.30 | 25.69 | 26.57 | OOM |
| Deep-HGCN            | 89.00 | 89.30 | **89.33** | 89.14 | 88.80 | OOM |
| MLP                  | 50.63 | 51.03 | 50.42 | **51.34** | 50.74 | 50.81 |
| Cora HyperGCN        | 48.38 | **50.75** | 26.37 | 26.36 | 23.29 | 19.63 |
| (co-c) HGNN           | **48.06** | 41.05 | 18.24 | 17.54 | 17.92 | 22.81 |
| Deep-HGCN            | 66.90 | 68.12 | 68.33 | 68.18 | 69.00 | **69.28** |
| MLP                  | 71.46 | 71.46 | **71.46** | 71.46 | 71.45 | 71.45 |
| PubmedHyperGCN       | 56.93 | **61.92** | 35.21 | 32.19 | 35.86 | 36.00 |
| (co-c) HGNN           | **42.72** | 39.52 | 36.32 | 36.43 | 36.06 | 36.37 |
| Deep-HGCN            | 74.08 | 74.42 | 74.10 | 74.51 | **74.68** | 74.63 |
| MLP                  | 52.00 | 51.92 | 51.75 | 51.75 | 51.65 | **52.00** |
| CiteseerHyperGCN     | 50.26 | **57.09** | 21.83 | 21.89 | 21.67 | 21.69 |
| (co-c) HGNN           | **39.50** | 37.06 | 19.68 | 20.32 | 19.22 | 20.42 |
| Deep-HGCN            | 61.00 | 60.90 | 61.85 | 62.10 | 61.69 | **62.51** |

The proof of Theorem 3 can be seen in supplementary materials. The theorem indicate the deep-HGCN replaces the fixed coefficients in SHGCN with the arbitrary coefficients, which can prevent the convergence of $P^K x$. Recall the section 3.2, we know the over-smoothing mainly cause by the fact that the node representation $P^K x$ in deep layer converges to a distribution that is independent of input node features. However, as suggest in Theorem 3, our model can carry the topology and initial node features information of the hypergraph to the deep, which not only enhance the heterogeneity of the node but also indeed relieve the problem of over-smoothing.

### 5 Experiments

In this section, we evaluate the performance of Deep-HGCN against the state-of-the-art hypergraph convolutional neural network models two classification task: hypernode classification on co-authorship and co-citation networks and visual object recognition.

### 5.1 Hypernode classification

**Datasets and Experiment settings** In this experiment, the task is semi-supervised node classification. We use the datasets provided by [Yadati et al., 2019] which include three co-citation network datasets: Cora, Pubmed and Citeseer, and two co-authorship network datasets: Cora, DBLP. We take the same train-test split as the released github\footnote{https://github.com/malllabiisc/HyperGCN} (i.e.10 different train-test splits), which different from [Yadati et al., 2019] reported on paper. Statistics of the datasets are summarized in Table 6, (refer to Appendix).
We use the Adam SGD optimizer to train the model for 300 epochs in total, with a learning rate of 0.01 and early stopping with a patience of 100 epochs. The feature dimension of the hidden layer is set as 32 and the dropout rate is set as 0.5. Experiments are done on a GeForce RTX 2080 Ti GPU and the results of baseline (MLP, HyperGCN, HGNN, HNHN, HGAT) are reproduced by their release Codes whose hyper-parameters follow the corresponding papers. We have used grid search to tune hyper-parameters $\alpha$ and $\beta$, the search range is provided in Table 9 in appendix.

**Comparison with SOTA** The results are shown in Table 2. We can observed that the Deep-HGCN outperform the baselines by a large margin in most cases. The main reason why the results of HyperGCN is different from the corresponding paper is that [Yadati et al., 2019] reported the results on 100 train-test split datasets.

![Figure 4: t-SNE visualization of the hypernode representations on Cora. Different colors represent different node classes. As the number of layers increases, the embeddings of different classes on HGNN also tend to be similar and lead to indistinguishable features. Instead, our model can get a more easily distinguishable node embedding.](image)

**Oversmoothing Analysis** Table 1 summaries the results with various depth on the five datasets. It is observed that HGNN and HyperGCN most achieve their best accuracy with 2 or 4 layers, and the accuracy drop rapidly when the number of layers exceeds 4, suggesting that they suffer from severe over-smoothing. Instead, our proposed Deep-HGCN achieve the best accuracy at layer 64, 8, 16, 32 and 64 respectively. It is noted that the performance of the Deep-HGCN consistently improves as we increase the number of layers, which verifies our analysis that our model can maintain the heterogeneity of node representation in deep layers.

We also draw t-SNE and feature maps for hypernode representations. Figure 4 is the t-SNE visualization of learned node representations. Figure 2 is the feature map of learned embedding of the first 50 hypernodes. Overall, the results suggest that HGCCNs suffer from severe over-smoothing problems, while Deep-HGCN can effectively relieve the over-smoothing problem and extend the HGNN into a truly deep model.

**Ablation study** In order to further explore the two techniques: *Initial residual* and *identity mapping*, we do the ablation study of them. As the Figure 3 show, compare to the HGNN, just adopting the identity mapping can effective alleviate the problem of over-smoothing and just applying the initial residual can greatly alleviate the over-smoothing. Using both of them at the same time increase the performance of the model to a greater extent. The result suggest that both techniques are need to overcome the problem of over-smoothing.

**5.2 Visual object recognition**

**Datasets and experimental settings** In this experiment, we compare with HGNN on two public objects: the Princeton ModelNet40 dataset and the National Taiwan University 3D model dataset NTU2012, where the hypergraph structure construction are as described in HGNN [Feng et al., 2019]. We use the same hype-parameters as [Feng et al., 2019]

**Results** The results on visual object recognition task are shown in Table 4. As shown in the results, the proposed Deep-HGCN can achieves much better performance compared with HGNN [Feng et al., 2019] on this two datasets. Detailed statistical information can refer to Table 7 in Appendix.

**Table 3: Summary of classification accuracy(%) results with various depths on object datasets.**

| Dataset | Method | 2 | 4 | 8 | 16 | 32 | 64 |
|---------|--------|---|---|---|----|----|----|
| NTU-2012 | HGNN | 83.16 | 77.59 | 5.36 | 5.20 | 5.28 | 5.25 |
|          | Deep-HGCN | 84.50 | 84.32 | 83.87 | 82.60 | 81.61 | 80.42 |
| Princeton | HGNN | 95.96 | 94.71 | 4.31 | 4.27 | 4.14 | 4.13 |
|          | Net40 Deep-HGCN | 96.08 | 96.00 | 94.54 | 95.22 | 95.20 | 88.24 |

Additional experiments results on the datasets given by HNHN [Dong et al., 2020] can be seen in the Appendix Table 5.
6 Conclusions

In this paper, we theoretically analyzed the over-smoothing problem of hypergraph convolutional neural networks from two perspectives: random walk on hypergraph and hypergraph Dirichlet energy. Specifically, the stationary distribution of random walk and the Dirichlet energy of hypernode embedding converging to zero both leads to over-smoothing. Therefore, we proposed a Deep-HGCN model with initial residual connection and identity mapping to avoid the over-smoothing issue. Furthermore, we gave theoretical justifications for Deep-HGCN’s effectiveness against over-smoothing by proving that a $k$-layer Deep-HGCN simulates a polynomial filter of order $k$ with arbitrary coefficients. Experiments show that the proposed model relieves the problem of over-smoothing and achieves state-of-the-art performance on the hypernode classification task.

References

Bai et al., 2019] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. arXiv preprint arXiv:1901.08150, 2019.

Chen and Wang, 2020] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. arXiv preprint arXiv:2006.13318, 2020.

Chen et al., 2021] Yuzhao Chen, Yatao Bian, Jiying Zhang, Xi Xiao, Tingyang Xu, Yu Rong, and Junzhou Huang. Diversified multiscale graph learning with graph self-correction. arXiv preprint arXiv:2103.09754, 2021.

Chen et al., 2022] Guanzi Chen, Jiying Zhang, Xi Xiao, and Yang Li. Graphheta: Test time adaptation on graph neural networks. arXiv preprint arXiv:2208.09126, 2022.

Ding et al., 2020] Kaize Ding, Jianlai Wang, Jundong Li, Dingcheng Li, and Huan Liu. Be more with less: Hypergraph attention networks for inductive text classification. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4927–4936, 2020.

Dong et al., 2020] Yihe Dong, Will Sawin, and Yoshua Bengio. Hhn: Hypergraph networks with hyperedge neurons. arXiv preprint arXiv:2006.12278, 2020.

Feng et al., 2019] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 3558–3565, 2019.

Gao et al., 2020] Yue Gao, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, and Changqing Zou. Hypergraph learning: Methods and practices. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.

He et al., 2016] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

Huang et al., 2010] Yuchi Huang, Qingshan Liu, Shaoting Zhang, and Dimitris N Metaxas. Image retrieval via probabilistic hypergraph ranking. In 2010 IEEE computer society conference on computer vision and pattern recognition, pages 3376–3383. IEEE, 2010.

Kim et al., 2011] Sungwoong Kim, Sebastian Nowozin, Pushmeet Kohli, and Chang Yoo. Higher-order correlation clustering for image segmentation. Advances in neural information processing systems, 24:1530–1538, 2011.

Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations (ICLR), 2017.

Klicpera et al., 2019] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Combining neural networks with personalized pagerank for classification on graphs. In International Conference on Learning Representations, 2019.

Li et al., 2018] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018.

Li et al., 2022] Fuyang Li, Jiying Zhang, Xi Xiao, bin zhang, and Dijun Luo. A simple hypergraph kernel convolution based on discounted markov diffusion process. In NeurIPS 2022 Workshop: New Frontiers in Graph Learning, 2022.

Min et al., 2020] Yimeng Min, Frederik Wenkel, and Guy Wolf. Scattering gc: Overcoming oversmoothness in graph convolutional networks. arXiv preprint arXiv:2003.08414, 2020.

Rong et al., 2019] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In International Conference on Learning Representations, 2019.

Sun et al., 2008] Liang Sun, Shuivang Ji, and Jieping Ye. Hypergraph spectral learning for multi-label classification. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 668–676, 2008.

Tan et al., 2014] Shulong Tan, Ziyu Guan, Deng Cai, Xuzhen Qin, Jiajun Bu, and Chun Chen. Mapping users across networks by manifold alignment on hypergraph. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 28, 2014.

Wu et al., 2019] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. arXiv preprint arXiv:1902.07153, 2019.

Xu et al., 2018] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping
knowledge networks. *arXiv preprint arXiv:1806.03536*, 2018.

[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. In *Advances in Neural Information Processing Systems*, pages 1511–1522, 2019.

[Yu et al., 2012] Jun Yu, Dacheng Tao, and Meng Wang. Adaptive hypergraph learning and its application in image classification. *IEEE Transactions on Image Processing*, 21(7):3262–3272, 2012.

[Zhang et al., 2022a] Jiying Zhang, Yuzhao Chen, Xi Xiao, Runiu Lu, and Shu-Tao Xia. Learnable hypergraph laplacian for hypergraph learning. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4503–4507. IEEE, 2022.

[Zhang et al., 2022b] Jiying Zhang, Fuyang Li, Xi Xiao, Tingyang Xu, Yu Rong, Junzhou Huang, and Yatao Bian. Hypergraph convolutional networks via equivalency between hypergraphs and undirected graphs, 2022.

[Zhang et al., 2022c] Jiying Zhang, Xi Xiao, Long-Kai Huang, Yu Rong, and Yatao Bian. Fine-tuning graph neural networks via graph topology induced optimal transport. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3730–3736. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.

[Zhao and Akoglu, 2020] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in {gnn}s. In *International Conference on Learning Representations*, 2020.

[Zhou et al., 2007] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *Advances in neural information processing systems*, pages 1601–1608, 2007.

[Zhou et al., 2020] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. Towards deeper graph neural networks with differentiable group normalization. *Advances in Neural Information Processing Systems*, 33, 2020.
Appendix

A Missing Proofs in Section 3.2

A.1 The proof of Lemma 1

\[ \tilde{\pi}P = 1^T D_v^{-\frac{1}{2}} D_v^{-\frac{1}{2}} H W D_v^{-\frac{1}{2}} H^T D_v^{-\frac{1}{2}} = (1^T D_v)(D_v^{-1} H W D_v^{-1} H^T) D_v^{-\frac{1}{2}} \]  

(12)

From Eq.(5), we know 

\[ (1^T D_v)(D_v^{-1} H W D_v^{-1} H^T) = 1^T D_v, \]

then we have

\[ \tilde{\pi}P = 1^T D_v D_v^{-\frac{1}{2}} = 1^T D_v^\frac{1}{2} = \tilde{\pi} \]  

(13)

A.2 the proof of Theorem 1.

Proof According to [Zhou et al., 2007], we can drive that the random walk defined on hypergraph have a unique stationary distribution ( proof by contradiction). Based on the Lemma 1, we know the random walk with transition matrix \( P \) has unique stationary distribution.

Let stationary distribution \( \pi \in \mathbb{R}^{|V| \times 1} \) and we have \( P\tilde{\pi} = \tilde{\pi} \),

\[ \lim_{k \to \infty} P^k = Q = [\tilde{\pi}, \tilde{\pi}, ..., \tilde{\pi}] \]  

(14)

where \( Q \in \mathbb{R}^{|V| \times |V|} \), and the each column in \( Q \) equal to \( \tilde{\pi} \). Then, it can derive that

\[ \lim_{k \to \infty} P^k x_{i,j} = Q^T x_{i,j} = \sum_{i=1}^{|V|} x_{ij} \tilde{\pi} \]  

(15)

B Missing Proofs in Section 3.3

Proof 1 (Lemma 2) Let us denote the eigenvalues of \( \Delta \) by \( \lambda_1, \lambda_2, ..., \lambda_{|V|} \), and the associated eigenvectors by \( v_1, v_2, ..., v_{|V|} \).

Suppose \( f = \sum_{i=1}^{|V|} c_i v_i \) where \( c_i \in \mathbb{R} \)

\[ E(f) = f^T \Delta f = f^T \sum_{i=1}^{|V|} c_i \lambda_i v_i = \sum_{i=1}^{|V|} c_i^2 \lambda_i \]  

(16)

Therefore

\[ E(Pf) = f^T (I_{|V|} - \Delta)^T \Delta (I_{|V|} - \Delta) f = f^T (I_{|V|} - \Delta) \Delta (I_{|V|} - \Delta) f \]

\[ = \sum_{i=1}^{|V|} c_i^2 \lambda_i (1 - \lambda_i)^2 \leq (1 - \lambda)^2 E(f) \]  

(2)

\[ E(X\Theta) = tr(\Theta^T X^T \Delta X \Theta) = tr(X^T \Delta X \Theta \Theta^T) \]

\[ \leq \|\Theta^T\|_2^2 tr(X^T \Delta X) = \|\Theta^T\|_2^2 E(X) \]  

(17)

\[ E(f) = f^T \Delta f = \frac{1}{2} \sum_{e \in E} \sum_{\{u,v\} \in V} w(e) h(u,e) h(v,e) \left( \frac{f(u)}{\sqrt{d(u)}} - \frac{f(v)}{\sqrt{d(v)}} \right)^2, \]  

(18)
\[ E(f) = f^T \Delta f \]
\[
= \frac{1}{2} \sum_{e \in E} \sum_{\{u,v\} \in V} \frac{w(e)h(u,e)h(v,e)}{\delta(e)} \left( \frac{f(u)}{\sqrt{d(u)}} - \frac{f(v)}{\sqrt{d(v)}} \right)^2 \\
\geq \frac{1}{2} \sum_{e \in E} \sum_{\{u,v\} \in V} \frac{w(e)h(u,e)h(v,e)}{\delta(e)} \left( \sigma \left( \frac{f(u)}{\sqrt{d(u)}} \right) - \sigma \left( \frac{f(v)}{\sqrt{d(v)}} \right) \right)^2
\]
\[ = E(\sigma(f)) \]

Then extending the above argument to vector field completes the proof.

Proof 2 (Theorem 2)

\[ E(X^{(l)}) = E(\sigma(PX^{(l-1)} \Theta^{(l-1)})) \leq E(PX^{(l-1)} \Theta^{(l-1)}) \leq \| \Theta^{(l-1)} \|_2^2 E(PX^{(l-1)}) = s_j E(PX^{(l-1)}) \leq s_j \lambda E(X^{(l-1)}) \]

C Missing Proofs in Section 4.2

C.1 The proof of Theorem 3

Proof. Taking the vertex features as graph signals, e.g., a column of the feature matrix \( X \) can be considered as a graph signal. For simplicity, we assume the signal vector \( x \in \mathbb{R}^d \) to be non-negative. Note that we can convert \( x \) into a non-negative input layer \( x^{(0)} \) by a linear transformation \( x^{(0)} = \Theta x \in \mathbb{R}^c \). We consider a weaker version of Deep-HGCN by fixing \( \alpha_l = 0.5 \) and fixing the weight matrix \((1 - \beta_l)I_c + \beta_l \Theta^{(l)}\) to be \( \gamma_l I_c \), where \( \gamma_l \) is a learnable parameter. We have

\[ x^{(l+1)} = \frac{1}{2} \sigma \left( P(x^{(l)} + x) \gamma_l I_c \right) \]

Since the input feature \( x \) is non-negative, we can remove coefficient \( \frac{1}{2} \) and ReLU operation for simplicity:

\[ x^{(l+1)} = \gamma_l P(x^{(l)} + x) = \gamma_l (I_c - \Delta)(x^{(l)} + x) \]

Consequently, we can express the final representation after \( K \) layers Deep-HGCN as:

\[ x^{(K+1)} = \left( \sum_{l=0}^{K-1} \left( \prod_{k=K-l-1}^{K-1} \gamma_k \right) (I_c - \Delta)^l \right) x. \]

On the other hand, as introduced in the beginning, a polynomial filter of \( K \)-layer Deep-HGCN can be expressed as:

\[ \left( \sum_{k=0}^{K-1} \theta_k \Delta^k \right) x = \left( \sum_{k=0}^{K-1} \theta_k (I_c - (I_c - \Delta))^k \right) x \\
= \left( \sum_{k=0}^{K-1} \theta_k \left( \sum_{l=0}^{k} (-1)^l \binom{k}{l} (I_c - \Delta)^l \right) \right) x \\
= \left( \sum_{l=0}^{K-1} \sum_{k=l}^{K-1} \theta_k (-1)^l \binom{k}{l} (I_c - \Delta)^l \right) x \]

To show that Deep-HGCN can express an arbitrary \( K \)-order polynomial filter, we need to prove that there exists a solution \( \gamma_l, l=0, \cdots, K-1 \) such that the corresponding coefficients of \((I_c - \Delta)^l \) in (13) and (14) are equivalent. More precisely, we need to show the following equation system has a solution \( \gamma_l, l=0, \cdots, K-1 \).

\[ \prod_{k=K-l-1}^{K-1} \gamma_k = \sum_{l=0}^{K-1} \theta_k (-1)^l \binom{k}{l}, k = 0, \cdots, K-1. \]
Note that we can solve the equation system by

\[ \gamma_{K-l-1} = \sum_{k=l}^{K-1} \theta_k (-1)^l \binom{k}{l} / \sum_{k=l-1}^{K-1} \theta_k (-1)^{l-1} \binom{k}{l-1} \]  

(26)

for \( l = 0, \ldots, K - 2 \) and \( \gamma_{K-1} = \sum_{k=0}^{K-1} \theta_k \).

D Missing Experiments in Section 5.2

D.1 Accuracy results

Visual object classification. See Table 4.

Citation network classification from HNHN. See Table 5.

Table 4: Summary of classification accuracy(\%) results with various depths on object datasets.

| Dataset     | Method      | Layers 2  | Layers 4  | Layers 8  | Layers 16 | Layers 32 | Layers 64 |
|-------------|-------------|----------|-----------|-----------|-----------|-----------|-----------|
| NTU2012     | HGNN        | 83.16±0.6 | 77.59±2.9 | 5.36±0.2  | 5.20±0.2  | 5.28±0.2  | 5.25±0.2  |
|             | Deep-HGCN   | 84.50±0.5 | 84.32±1.0 | 83.87±0.6 | 82.60±0.8 | 81.61±0.9 | 80.42±1.1 |
| ModelNet40  | HGNN        | 95.96±0.5 | 94.71±0.9 | 4.31±0.2  | 4.27±5.1  | 4.14±0.1  | 4.13±0.1  |
|             | Deep-HGCN   | 96.08±0.4 | 96.00±0.5 | 94.54±2.3 | 95.22±0.9 | 95.20±0.5 | 88.24±15  |

Table 5: Summary of classification accuracy(\%) results of the experiment on the datasets that provide by HNHN

| Dataset     | Method      | Layers 2  | Layers 4  | Layers 8  | Layers 16 | Layers 32 | Layers 64 |
|-------------|-------------|-----------|-----------|-----------|-----------|-----------|-----------|
| Cora        | HGNN        | 57.67±0.9 | 51.87±3.1 | 42.64±2.8 | 40.37±0.05| 40.38±0.07| 40.42±0.6 |
|             | HNHN [Dong et al., 2020] | 55.06±4.1 | 35.15±2.7 | 34.54±6.1 | 15.56±10.9| 4.54±0.7  | 10.70±3.5 |
|             | Deep-HGCN   | 58.89±0.7 | 58.92±0.4 | 59.45±0.3 | 57.60±0.9 | 61.19±0.3 | 60.97±0.7 |
| Citeseer    | HGNN        | 65.67±0.7 | 63.63±2.2 | 37.28±12.7| 21.76±0.2 | 21.45±0.5 | 21.80±0.3 |
|             | HNHN        | 64.68±5.2 | 43.66±5.7 | 27.64±2.4 | 26.27±4.6 | 17.94±4.2 | 17.94±4.2 |
|             | Deep-HGCN   | 65.72±0.9 | 67.62±0.9 | 66.50±1.4 | 67.09±1.3 | 67.53±1.1 | 67.89±0.5 |

D.2 Datasets

See Table 6,7, 8.

Table 6: Real-world hypergraph datasets used in our work.(co-a represents co-authorship, and co-c represent co-citation)

| Dataset      | Hypernodes | Hyperedges | Features | Classes |
|--------------|------------|------------|----------|---------|
| Cora (co-a)  | 2708       | 1072       | 1433     | 7       |
| DBLP (co-a)  | 43413      | 22535      | 1425     | 6       |
| Pubmed (co-c)| 19717      | 7963       | 500      | 3       |
| Cora (co-c)  | 2708       | 1579       | 1433     | 7       |
| Citeseer (co-c) | 3312   | 1079       | 3703     | 6       |

D.3 Hyper-parameter search strategy

See Table 9.
Table 7: The detailed information of the ModelNet40 and NTU datasets.

| Dataset       | ModelNet40 | NTU2012 |
|---------------|------------|---------|
| Objects       | 12311      | 2012    |
| MVCNN Feature | 4096       | 4096    |
| GVCNN Feature | 2048       | 2048    |
| Training node | 9843       | 1639    |
| Testing node  | 2468       | 373     |
| Classes       | 40         | 67      |

Table 8: Datasets used in the experiment compared with HNHN.

| Dataset | Hypernodes | Hyperedges | Features | Classes |
|---------|------------|------------|----------|---------|
| Cora    | 16313      | 7389       | 1000     | 10      |
| Citeseer| 1498       | 1107       | 3703     | 6       |

Table 9: Hyper-parameter search range for Deep-HGCN model.

| Hyper-parameter | Range                        |
|-----------------|------------------------------|
| $\alpha$        | {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1} |
| $\lambda$       | {0, 0.5, 1.0, 1.5, 2.0, 2.5}  |