Hypergraph Convolutional Network with Hybrid Higher-Order Neighbors

Jiahao Huang[1,2], Fangyuan Lei[1,2(✉)], Senhong Wang[3], Song Wang[4], and Qingyun Dai[1,2]

[1] Guangdong Provincial Key Laboratory of Intellectual Property and Big Data, Guangdong Polytechnic Normal University, Guangzhou 510665, China
[2] School of Electronic and Information, Guangdong Polytechnic Normal University, Guangzhou 510665, China
[3] School of Information Engineering, Guangdong University of Technology, Guangzhou 510006, China
[4] Department of Computer Science and Engineering, University of South Carolina, Columbia, SC 29208, USA

Abstract. Hypergraph-based methods can learn non-pairwise associations more efficiently in many real-world datasets. However, existing hypergraph-based methods do not consider the relationship of the hybrid neighborhood. To address this issue, we propose a hybrid higher-order neighborhood based hypergraph convolutional network (HybridHGCN). Technically, feature embeddings are generated via k-hop hypergraph convolution layers and mixed by the hybrid message operator. To evaluate the proposed HybridHGCN, we conduct experiments on the citation network datasets and the visual object datasets. The experimental results show that HybridHGCN brings significant improvements over state-of-the-art hypergraph neural network baselines.

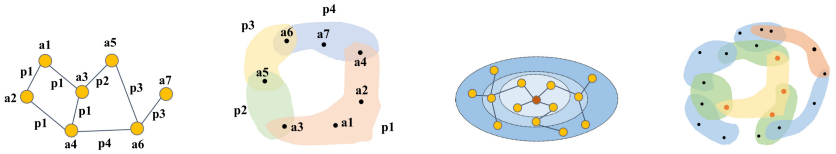
Keywords: Hypergraph · Higher-order correlation · Hypergraph convolutional networks

1 Introduction

Graphs are widely used to model the pair-wise relationships in the real world, such as collaboration networks and co-authoring networks [5,18]. An example of such a graph is shown in Fig. 1(a), in which $a1$,..., to $a7$ are the nodes which represent the authors, and $p1$,..., to $p4$ are the edges which represent the papers connecting the co-authors. Non-pair-wise and complex relationships among entities can be more effectively and flexibly modeled by a hypergraph [6]. An example of a hypergraph is shown in Fig. 1(b), in which $p1$,..., to $p4$ are the hyperedges that connect the multiple co-authors. The number of nodes connected by a hyperedge is defined as the degree of the hyperedge. A hypergraph is simplified to a graph when the degree of the hyperedge is set to 2.

H. Ma et al. (Eds.): PRCV 2021, LNCS 13022, pp. 103–114, 2021.
https://doi.org/10.1007/978-3-030-88013-2_9

Graph convolutional network (GCN) [15] has been applied to citation networks and knowledge graphs, with significant improvement in the semi-supervised node classification task. Although GCN can be used for classification tasks in visual data and social networks, it is highly desired to extend it to hypergraph structure given its more flexible modeling of complex relationships in many real-world networks compared with graphs. Recently, many researchers explored the hypergraph application in GNNs [3,4,10,14,25]. Feng et al. [10] proposed a hypergraph neural network (HGNN) by using hypergraph Laplacian to learn the message of non-pair-wise relationships. Yadati et al. [25] simplified the hypergraph to a graph and then applied the GCN. However, existing hypergraph-based neural networks do not consider the k-hop neighbor relations.



(a) Simple graph (b) Hypergraph (c) K-hop graph (d) K-hop hypergraph

Fig. 1. Illustration of the graph/hypergraph structures, higher-order graph and higher-order hypergraph. (a) A simple graph represents the pair-wise relations among authors and papers. (b) A hypergraph with 4 hyperedges that represent the non-pair-wise relations among authors and papers. (c) A k-hop graph where ovals of different colors represent different orders. (d) A k-hop hypergraph where hyperedges of different colors represent different orders (Color figure online).

Inspired by [1], to consider the k-hop neighbor in a hypergraph, we propose a hybrid higher-order neighborhood based hypergraph convolutional network (HybridHGCN). To aggregate k-hop neighbor messages, HybridHGCN is composed of k-hop hypergraph convolutional layers and a hybrid message operator. As the higher-order neighbors of a hypergraph can be described by the incidence matrix of higher-order power. An example of a k-hop graph is shown in Fig. 1(c), in which ovals of different colors represent different orders. As illustrated in Fig. 1(d), a k-hop hypergraph is represented by different-order hyperedges, shown in different colors. In the incidence matrix of different power, the receptive field of neighbors is broadened as the order increases. Updated by k-hop hypergraph convolutional layers, the generated features embedding represent the k-hop neighbor messages. In the hybrid message operator, we use an element-wise max pooling operator to mix k-hop neighbor messages. To evaluate the proposed HybridHGCN, we construct three different hop hypergraph convolutional layers and examine the node classification performance on both citation network datasets and visual object datasets.

The main contributions of this paper are summarized as follows.

– We propose HybridHGCN, a new method to capture higher-order and low-order neighbor relations and it enhance the representation capability of the hypergraph network.
– We propose the hypergraph structuration with the higher-order incidence matrix to broaden the receptive field of the hypergraph network.
– The experimental results show the proposed HybridHGCN achieve the new state-of-the-art performance on citation networks and visual object datasets.

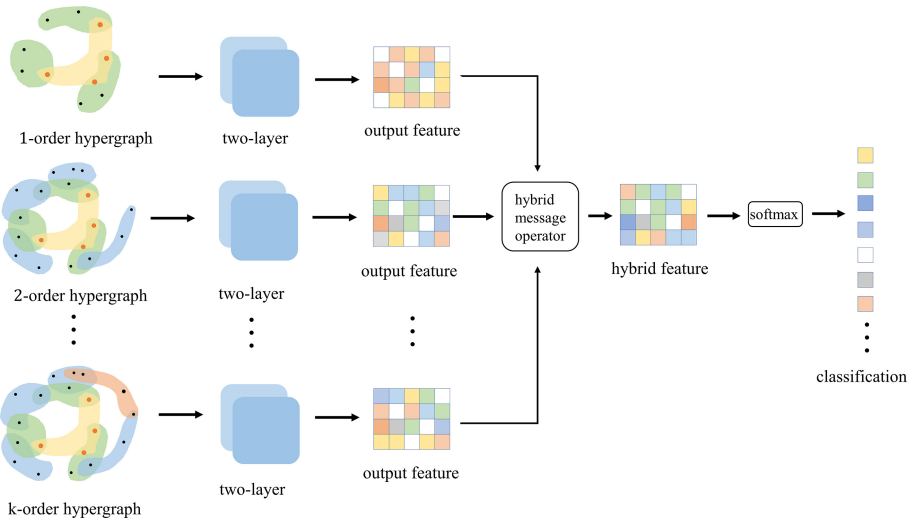


Fig. 2. Overview of the proposed hybrid higher-order neighborhood based hypergraph convolutional network (HybridHGCN). The parameters 1,..., to k denote the hypergraphs with different hops. The two-layer denotes the adopted two hypergraph convolutional layers. The output feature embedding is composeed of N-nodes with D-dimension features. Mixing by a hybrid message operator, the number of nodes and dimension of features remain unchanged.

2 Related Works

2.1 Graph Neural Networks

Due to the excellent performance of deep neural networks on structured data from various tasks, Bronstein et al. [7] extended the neural network model to the graph structure data drawn from non-Euclidean space. Kipf et al. [15] proposed Graph Convolutional Network (GCN) by learning neighboring node representations through the convolution defined by the graph Laplacian. To solve the problem of GCN's dependence on the global graph structure, Veličković et al. [22] proposed Graph Attention Network (GAT) in which the attention mechanism assigns different weights to different nodes. Abu-El-Haija et al. [1]

proposed the MixHop model to explore neighborhood mixing relationships represented by repeatedly mixing feature representations. More details about the current research of graph neural networks and graph representation can be found in excellent surveys [9,23,29].

2.2 Learning on Hypergraph

On hypergraphs, clique expansion or star expansion [2,16,28] are widely used to mine the hypergraph structure. Su et al. [21] utilized a hypergraph structure to represent the correlation between different objects in a multi-view 3D dataset and then proposed a vertex-weighted hypergraph classification algorithm. Yu et al. [26] proposed an alternating optimization method to optimize the label and hyperedge weights in the hypergraph learning process. Hayashi et al. [12] employed random walks based on edge-related vertex weights to construct different hypergraph Laplacian matrices, and proposed a flexible hypergraph structure data clustering framework.

2.3 Hypergraph Neural Network

Feng et al. [10] proposed Hypergraph Neural Network (HGNN) to model the non-pair-wise relations as the weighted hypergraph. The hypergraph neural networks was also applied to visual classification [19]. Jiang et al. [13] proposed the dynamic hypergraph neural network by extending the dynamic hypergraph learning [27]. Bandyopadhyay et al. [4] proposed the line hypergraph convolutional network by mapping the hypergraph to a weighted attribute line graph.

3 Preliminary Knowledge

3.1 Hypergraph

In this section, we briefly introduce the preliminary knowledge of hypergraphs. An edge in a simple graph connects two vertices. Compared with simple graph, a weighted hypergraph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is a set of n vertices, $\mathcal{E} = \{e_1, e_2, ..., e_m\}$ is a set of m hyperedges and $\mathcal{W} = \{(w(e_1), w(e_2), ..., w(e_m)\}$ is a set of hyperedge weight. The structure of hypergraph $\mathcal{G}$ can be represented by an incidence matrix $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$ with entries

$$h(v, e) = \begin{cases} 0 \text{ if } v \notin e, \\ 1 \text{ if } v \in e. \end{cases} \tag{1}$$

In practice, $h(v, e)$ takes continuous values in the range of $[0, 1]$ which indicates the importance of the vertex v for hyperedge e. The degree of a vertex $v \in \mathcal{V}$ is defined by

$$d(v) = \sum_{e\in\mathcal{E}} w(e)h(v, e), \tag{2}$$

where e denotes a hyperedge in the hyperedge set $\mathcal{E}$, and the degree of hyperedge $e \in \mathcal{E}$ is defined by

$$\delta(e) = \sum_{v \in \mathcal{V}} h(v, e). \tag{3}$$

Let $\mathbf{D}_v$ be the diagonal matrix of the vertex degrees, $\mathbf{D}_e$ be the diagonal matrix of the hyperdege degrees, and $\mathbf{W}$ be the diagonal matrix of the hyperedge weights, i.e., $\mathbf{D}_v = diag(d(v_1), d(v_2), ..., d(v_n))$, $\mathbf{D}_e = diag(\delta(e_1), \delta(e_2), ..., \delta(e_m))$, and $\mathbf{W} = diag(w(e_1), w(e_2), ..., w(e_m))$.

3.2 Hypergraph Convolution Network

For a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ and hypergraph signal $\mathbf{X}$, the hypergraph convolution layer $f(\mathbf{X}, \mathbf{W}, \boldsymbol{\Theta})$ based on the theory of spectral convolution on hypergraph is defined as

$$\mathbf{X}^{(l+1)} = \sigma(\mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-\frac{1}{2}}\mathbf{X}^{(l)}\Theta^{(l)}), \tag{4}$$

where $\mathbf{X}^{(l)} \in \mathbb{R}^{N \times D^{(l)}}$ represents the N vertex features in the $l-th$ layer, $\Theta^{(l)}$ is a learnable parameter of filter in the $l-th$ layer and $\sigma(\cdot)$ denotes the nonlinear activation function. Hyperedge weight $\mathbf{W} \in \mathbb{R}^{N \times N}$ reflects the importance of different hyperedges in the network. A two-layer hypergraph convolutional network model can be defined by

$$\mathbf{Y} = softmax(\hat{\mathbf{H}}(\sigma(\hat{\mathbf{H}}\mathbf{X}^{(1)}\Theta^{(1)}))\Theta^{(2)})), \tag{5}$$

where $\hat{\mathbf{H}} = \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^T\mathbf{D}_v^{-\frac{1}{2}}$, and $\Theta^{(1)}$ and $\Theta^{(2)}$ represent the trainable parameter matrix in first layer and second layers respectively.

4 Method

In this section, we propose the HybridHGCN to capture k-hop neighbor relations. As shown in Fig. 2, HybridHGCN consists of k branches and a hybrid message operator that each branch takes the two-layer structure. We employ incidence matrices of different order of powers to represent the k-hop hypergraph. In the hypergraph convolutional layer, node representation is updated with the k-hop neighbor message by the incidence matrix of k-order power. To represent different-order neighbor relations, we propose a hybrid message operator to mix feature embeddings generated from each branch. Specifically, the hybrid message operator is implemented before the $softmax$ layer. The HybridHGCN consisting of k branches and a hybrid message operator can be written as

$$Y = softmax(HM(\sigma(\hat{H}^{(1)}X^{(l)}\Theta^{(l)}), \sigma(\hat{H}^{(2)}X^{(l)}\Theta^{(l)}), ..., \sigma(\hat{H}^{(k)}X^{(l)}\Theta^{(l)})), \tag{6}$$

where $HM(\cdot)$ is a hybrid message operator, $\hat{H} = D_v^{-\frac{1}{2}} H W D_e^{-1} H^T D_v^{-\frac{1}{2}}$, and $\sigma(\cdot)$ is the Relu activation function. The parameter k denotes the power of the incidence matrix, and l represents the layer of the hypergraph convolution. In $HM(\cdot)$ layer, the input feature of $(l+1)$-layer is the output of l-layer hypergraph convolution. When parameter $l = 1$, the input feature of the first hypergraph convolutional layer is the initial feature X. Specifically, $\hat{H}^{(k)}$ represents the incidence matrix of k-th order of power. The diagonal matrices of the edge degrees $D_e^{(k)}$ and vertex degrees $D_v^{(k)}$ correspond to the k-th order of hypergraph Laplacian $\hat{H}^{(k)}$, respectively. In the process of simultaneous high order and low order message propagation, we propose a hybrid message operator $HM(\cdot)$ as follows:

$$Out_{HM} = HM(Fts_{H^{(1)}}, Fts_{H^{(2)}}, ..., Fts_{H^{(k)}}), \tag{7}$$

where Out_{HM} represents the mixed feature. The features $Fts_{H^{(1)}}$, $Fts_{H^{(2)}}$ and $Fts_{H^{(k)}}$ represent the output features of hypergraph convolution layers associated with the $1st$, $2nd$ and $k-th$ orders of power incidence matrices respectively. $HM(\cdot)$ performs an element-wise operation on eigenvector matrices of different orders, which selects the max elements. In this paper, we set the parameter $l = 2$ which means the HybridHGCN has two layers and we have

$$\begin{aligned} Y = softmax(HM(\hat{H}^{(1)}\sigma(\hat{H}^{(1)}X^{(l_1)}\Theta^{(l_1)})\Theta^{(l_2)}, \hat{H}^{(2)}\sigma(\hat{H}^{(2)}X^{(l_1)}\Theta^{(l_1)})\Theta^{(l_2)}, \\ ..., \hat{H}^{(k)}\sigma(\hat{H}^{(k)}X^{(l_1)}\Theta^{(l_1)})\Theta^{(l_2)})), \end{aligned} \tag{8}$$

where $\hat{H}$ is the normalized hypergraph Laplancian, $\Theta^{(l_1)} \in \mathbb{R}^{d_0 \times d_1}$ and $\Theta^{(l_2)} \in \mathbb{R}^{d_1 \times d_2}$ are the learnable weight parameters. For classification with q classes, we adopt cross entropy as the loss function of HybridHGCN:

$$\mathcal{L} = -\sum_{i \in \mathcal{V}_L} \sum_{j=1}^{q} \hat{Y}_{ij} \ln Y_{ij}, \tag{9}$$

where $\mathcal{V}_L$ denotes the set of labeled examples and q represents the number of classes. $\hat{Y}_{ij}$ denotes the actual label of a node in $\mathcal{V}_L$ and Y_{ij} is the predicted label computed by Eq. (8).

5 Experiments

5.1 Datasets and Baseline

In our experiments, we use classification accuracy as the evaluation criteria. To evaluate the HybridHGCN, we employ five datasets including citation networks and visual objects. Cora, Citeseer and Pumbed [17] are publicly used citation network datasets, which contain the graph structures of citations with the bags-of-words feature vector of documents. In Cora, Citeseer, and Pumbed, nodes correspond to documents and edges represent the relation between each pair of nodes. Specifics of Cora, Citeseer, and Pumbed are presented in Table 1.

Table 1. Summary of citation network datasets.

Dataset	Cora	Citeseer	Pumbed
Number of nodes	2,708	3,327	19,717
Number of edges	5,278	4,552	44,324
Length of features	1,433	3,703	500
Number of classes	7	6	3

The Priceton ModelNet40 dataset [24] contains 12,311 visual objects of 40 categories including airplane, bathtub, car, dresser, guitar, and so on. The National Taiwan University 3-D model (NTU) dataset [8] contains 2,012 visual objects of 67 categories, including chair, clock, door, frame and so on. The summary of visual object, categories, and length of MVCNN and GVCNN extracted features are given in Table 2.

Table 2. Summary of visual object datasets.

Dataset	*ModelNet*40	*NTU*
Number of object	12,311	2,012
Length of MVCNN features	4,096	4,096
Length of GVCNN features	2,048	2,048
Number of classes	40	67

We compare the following methods in our experiments. We denote HGNN as HGNN-s when HGNN adopt the same graph structure as HyperGCN [25] and LHCN [4].

- HyperGCN [25]: The method approximates the hypergraph Laplacian to graph and then performs graph convolution. 1-HyperGCN and FastHyperGCN are two variants of HyperGCN. The former approximates the hyperedge by a pair of edges and adds mediators to enhance the performance. The latter reduces computation time where the hypergraph Laplacian is computed only once before training.
- Line Hypergraph Convolution Network (LHCN) [4]: The hypergraph structure is mapped to an attributed and weighted line graph which adapts in graph convolution.
- Hypergraph Neural Network (HGNN) [10]: The method adopts the normalized hypergraph Laplacian to perform graph convolution in weighted clique expansion hypergraph.

5.2 Experimental Setting

For citation networks and visual object datasets, we construct the hypergraph from the original graph structure and Euclidean distance of visual objects,

respectively. For each node in the graph of citation networks, we take it as a central node and set a hyperedge with this central node as the centroid. The other nodes that connect to the centroid are added into the hyperedge, and the entries of hypergraph structure are formulated as Eq. (1). HGNN [10] and HybridHGCN adopt the same hypergraph structure. The graph structures of HyperGCN have 1,579, 1,079 and 7,963 edges for Cora, Citeseer and Pubmed respectively.

For visual object datasets, we select Mutil-view CNN [20] and Group-view CNN [11] to extract features, which have shown excellent performance in the representation of 3D shape. Considering the feature of each visual object, we calculate its Euclidean distance with other objects, and then select the k nearest nodes to connect. In our experiments, we set the hyper-parameter k to be 10. The entries of the constructed hypergraph structure is as follows.

$$h(v,e) = \begin{cases} 0 & \text{if } v \notin e \\ exp\left(-\frac{2D_{ij}^2}{\Delta}\right) & \text{if } v \in e, \end{cases} \tag{10}$$

where D_{ij} denotes the Euclidean distance between node i and node j, and Δ denotes the average pairwise distance of all nodes.

For Cora, Citeseer, and Pubmed datasets, the dimensions of the hidden units of HybridHGCN are 32, 16, and 64com, respectively, and the orders of the incidence matrices are 1, 2, and 3 respectively.

For ModelNet40 and NTU datasets, the dimensions of the hidden units of HybridHGCN are 256, and the orders of the incidence matrices are 1, 2, and 3 respectively.

5.3 Experimental Results and Discussion

For citation networks datasets, experiment results are shown in Table 3, where performances of comparison methods are directly taken from [4,25]. The proposed HybridHGCN achieves the best testing accuracy of 81.88%, 70.96%, and 78.31% on Cora, Citeseer, and Pubmed respectively. We adopt the datasets split by using 5.2%, 4.1%, and 0.3% of data for training respectively and the rest is for testing. The train-test split is the same as HyperGCN [25] which samples the nodes of the same size from each class.

We also evaluate the specific effects of different label rates on HGNN and HybridHGCN for the citation datasets, by trying to use 5%, 10%, 20%, 30%, and 40% of data for training, respectively and the rest is for validating and testing. As shown in Table 4, compared with HGNN, the accuracy of HybridHGCN increases by 0.1% to 1.79% for different data splits on Cora, with the more obvious improvement on the low label rate. For Citeseer, as shown in Table 5, HybridHGCN shows an increase in accuracy of 0.3% to 2.0% for different data splits compared to HGCN. It's worth noting that HybridHGCN improves performance even more when the label rates are high in Citeseer. For Pubmed, from Table 6 we can see that HybridHGCN achieves 0.2% to 0.3% increase in accuracy for different data splits, which are slight performance improvements over

Table 3. Test accuracy (%) on citation networks classification. ± represents the standard deviation.

Method	Cora	Citeseer	Pubmed
HGNN-s [25]	67.59 ± 1.8	62.60 ± 1.6	70.59 ± 1.5
1-HyperGCN [25]	65.55 ± 2.1	61.13 ± 1.9	69.92 ± 1.5
FastHyperGCN [25]	67.57 ± 1.8	62.58 ± 1.7	70.52 ± 1.6
HyperGCN [25]	67.63 ± 1.7	62.65 ± 1.6	74.44 ± 1.6
LHCN [4]	73.34 ± 1.7	63.19 ± 2.2	70.76 ± 2.4
HGNN [10]	80.99 ± 0.8	69.72 ± 1.0	78.14 ± 1.7
HybridHGCN	**81.88 ± 0.2**	**70.96 ± 0.8**	**78.31 ± 1.6**

Table 4. Test accuracy (%) on Cora dataset classification. ± represents the standard deviation. 5% to 40% represent the percentage of data used for training.

Method	Cora				
	5%	10%	20%	30%	40%
HGNN [10]	79.86 ± 1.1	81.72 ± 1.6	84.60 ± 0.9	86.36 ± 1.4	88.02 ± 1.4
HybridHGCN	**81.04 ± 0.5**	**83.28 ± 0.8**	**85.20 ± 1.1**	**87.50 ± 0.9**	**89.81 ± 0.4**

HGNN. The success of HybridHGCN lies in the addition of the k-hop neighbor message to improve node representation capabilities.

For the experiment on ModelNet40 and NTU datasets, we follow the same 80%–20% dataset split for training and testing respectively as in [24]. For visual object datasets, from Table 7 we can observe that HybridHGCN performs better than the rest under most CNNs features. On the ModelNet40 dataset, compared with the MVCNN features and GVCNN features, the performance of HybridHGCN is close to HGNN. On the NTU dataset, HybridHGCN brings the gain of 2.24% and 2.93% on MVCNN features and GVCNN features respectively. The results indicate that HybridHGCN is more effective in node classification tasks than HGNN.

Table 5. Test accuracy (%) on Citeseer dataset classification. ± represents the standard deviation. 5% to 40% represent the percentage of data used for training.

Method	Citeseer				
	5%	10%	20%	30%	40%
HGNN [10]	70.22 ± 1.2	71.70 ± 0.6	**73.82 ± 1.0**	74.98 ± 0.5	75.04 ± 1.3
HybridHGCN	**70.58 ± 1.1**	**72.32 ± 0.6**	73.60 ± 0.5	**76.08 ± 1.4**	**77.02 ± 1.4**

Table 6. Test accuracy (%) on Pubmed dataset classification. ± represents the standard deviation. 5% to 40% represent the percentage of data used for training.

Method	Pubmed				
	5%	10%	20%	30%	40%
HGNN [10]	83.16 ± 0.2	83.80 ± 0.8	**84.54 ± 0.5**	84.60 ± 0.3	84.66 ± 0.5
HybridHGCN	**83.38 ± 0.3**	**84.12 ± 0.3**	84.22 ± 0.1	**84.96 ± 0.3**	**84.90 ± 0.6**

Table 7. Test accuracy (%) on ModelNet40 and NTU datasets classification. ± represents the standard deviation. MVCNN and GVCNN represent the features extracted from Multi-view CNN [20] and Group-view CNN [11] respectively.

Method	ModelNet40		NTU	
	MVCNN	GVCNN	MVCNN	GVCNN
HGNN [10]	**91.00**	92.60	75.60	82.50
HybridHGCN	90.62 ± 0.0	**92.76 ± 0.1**	**77.84 ± 0.1**	**85.43 ± 0.1**

6 Conclusions

This paper proposed a hybrid higher-order neighborhood based hypergraph convolutional network (HybridHGCN), which explores the k-hop neighbor message. We conducted extensive experiments on hypergraphs construct on citation networks and visual object datasets, and the results showed that HybridHGCN performs better than the state-of-the-art methods. For the future work, we plan to further reduce the feature redundancy in mixing the k-hop neighbor messages.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China under Grant U1701266, in part by Guangdong Provincial Key Laboratory of Intellectual Property and Big Data under Grant 2018B030322016, in part by Special Projects for Key Fields in Higher Education of Guangdong under Grant 2020ZDZX3077 and Grant 2021ZDZX1042, and in part by Qingyuan Science and Technology Plan Project under Grant 170809111721249 and Grant 170802171710591.

References

1. Abu-El-Haija, S., et al.: Mixhop: higher-order graph convolutional architectures via sparsified neighborhood mixing. In: Proceedings of the International Conference on Machine Learning, pp. 21–29 (2019)
2. Agarwal, S., Branson, K., Belongie, S.: Higher order learning with graphs. In: Proceedings of the International Conference on Machine Learning, pp. 17–24 (2006)
3. Bai, S., Zhang, F., Torr, P.H.S.: Hypergraph convolution and hypergraph attention. Pattern Recogn. **110**, 107637 (2021)
4. Bandyopadhyay, S., Das, K., Narasimha Murty, M.: Line hypergraph convolution network: Applying graph convolution for hypergraphs (2020). arXiv preprint: arXiv:2002.03392
5. Benson, A.R., Gleich, D.F., Leskovec, J.: Higher-order organization of complex networks. Science **353**(6295), 163–166(2016)

6. Bretto, A.: Hypergraph Theory. An Introduction. Mathematical Engineering. Springer, Cham (2013)
7. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond Euclidean data. IEEE Signal Process. Mag. **34**(4), 18–42 (2017)
8. Chen, D., Tian, X., Shen, Y., Ouhyoung, M.: On visual similarity based 3d model retrieval. Comput. Graph. Forum **22**, 223–232 (2003)
9. Chen, F., Wang, Y., Wang, B., Jay Kuo, C.-C.: Graph representation learning: a survey. APSIPA Trans. Signal Inf. Process. 9 (2020)
10. Feng, Y., You, H., Zhang, Z., Ji, R., Gao, Y.: Hypergraph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, pp. 3558–3565 (2019)
11. Feng, Y., Zhang, Z., Zhao, X., Ji, R., Gao, Y.: GVCNN: group-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 264–272 (2018)
12. Hayashi, K., Aksoy, S.G., Park, C.H., Park, H.: Hypergraph random walks, laplacians, and clustering. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 495–504 (2020)
13. Jiang, J., Wei, Y., Feng, Y., Cao, J., Gao, Y.: Dynamic hypergraph neural networks. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 2635–2641 (2019)
14. Kim, E.-S., Kang, W.Y., On, K.-W., Heo, Y.-J., Zhang, B.-T.: Hypergraph attention networks for multimodal learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14581–14590 (2020)
15. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2016). arXiv preprint: arXiv:1609.02907
16. Li, P., Faltings, B.: Hypergraph learning with hyperedge expansion. In: Flach, P.A., De Bie, T., Cristianini, N., (eds.) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2012. LNCS, vol. 7523, pp. 410–425. Springer, Berlin (2021). https://doi.org/10.1007/978-3-642-33460-3_32
17. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93 (2008)
18. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation (2018). arXiv preprint: arXiv:1811.05868
19. Shi, H., et al.: Hypergraph-induced convolutional networks for visual classification. IEEE Trans. Neural Netw. Learn. Syst. **30**(10), 2963–2972 (2018)
20. Hang, S., Subhransu, M., Evangelos, K., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE International Conference On Computer Vision, pp. 945–953 (2015)
21. Su, L., Gao, Y., Zhao, X., Wan, H., Gu, M., Sun, J.: Vertex-weighted hypergraph learning for multi-view object classification. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 2779–2785 (2017)
22. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks (2017). arXiv preprint: arXiv:1710.10903
23. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. **32**, 4–24 (2020)
24. Wu, Z., et al.: 3d shapenets: a deep representation for volumetric shapes. In: Proceedings of the IEEE Conference On Computer Vision And Pattern Recognition, pp. 1912–1920 (2015)

25. Yadati, N., et al.: HyperGCN: a new method of training graph convolutional networks on hypergraphs (2018). arXiv preprint: arXiv:1809.02589
26. Jun Yu, Dacheng Tao, and Meng Wang.: Adaptive hypergraph learning and its application in image classification. IEEE Transactions on Image Processing, vol. 21, no. 7, pp. 3262–3272(2012)
27. Zhang, Z., Lin, H., Gao, Y.: KLISS BNRist.: dynamic hypergraph structure learning. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 3162–3169 (2018)
28. Zhou, D., Huang, J., Schölkopf, B.: Learning with hypergraphs: clustering, classification, and embedding. Adv. Neural Inf. Process. Syst. **19**, 1601–1608 (2006)
29. Zhou, J., et al.: Graph neural networks: a review of methods and applications (2018). arXiv preprint: arXiv:1812.08434