LEARNABLE HYPERGRAPH LAPLACIAN FOR HYPERGRAPH LEARNING

Jiying Zhang[1,*]   *Yuzhao Chen*[1,*]   *Xi Xiao*[1†]   *Runiu Lu*[1]   *Shu-Tao Xia*[1,2]

[1] Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
[2] Research Center of Artificial Intelligence, Peng Cheng Laboratory, Shenzhen, China

{zhangjiy20, chen-yz19}@mails.tsinghua.edu.cn, {xiaox,lurn,xiast}@sz.tsinghua.edu.cn

ABSTRACT

HyperGraph Convolutional Neural Networks (HGCNNs) have demonstrated their potential in modeling high-order relations preserved in graph-structured data. However, most existing convolution filters are localized and determined by the pre-defined initial hypergraph topology, neglecting to explore implicit and long-range relations in real-world data. In this paper, we propose the *first* learning-based method tailored for constructing adaptive hypergraph structure, termed HypERgrAph Laplacian aDaptor (HERALD), which serves as a generic plug-and-play module for improving the representational power of HGCNNs. Specifically, HERALD adaptively optimizes the adjacency relationship between vertices and hyperedges in an end-to-end manner and thus the task-aware hypergraph is learned. Furthermore, HERALD employs the self-attention mechanism to capture the non-local paired-nodes relation. Extensive experiments on various popular hypergraph datasets for node classification and graph classification tasks demonstrate that our approach obtains consistent and considerable performance enhancement, proving its effectiveness and generalization ability.

Index Terms— hypergraph convolutional neural network, adaptive hypergraph structure, self-attention, non-local relation

1. INTRODUCTION

Recently, Graph Convolutional Neural Networks (GCNNs) have been developed for various learning tasks on graph data [1, 2], such as social networks [3], citation networks [4] and biomedical networks [5]. GCNNs have shown superiority on graph representation learning compared with traditional neural networks used to process regular data.

Many graph neural networks have been developed to model simple graph whose edge connects exactly two vertices. In the meantime, more and more researchers noted that the data structure in practice could be beyond pair connections, and intuitive pairwise connections among nodes are usually insufficient for capturing higher-order relationships.

Hypergraphs, generalizations of simple graphs, have a powerful ability to model complex relationships in the real world, since their edges can connect any number of vertices [6–8]. For example, in a co-citation relationship [9], papers act as vertices, and citation relationships become hyperedges. Consequently, a new research area aiming at establishing convolutional networks on hypergraphs attracts a surge of attention. Existing representative works in the literature [9–12] which have designed basic network architecture for hypergraph learning. However, a major concern is that these methods are built upon an intrinsic hypergraph and not capable of dynamically optimizing the hypergraph topology. The topology plays an essential role in the message passing across nodes, and its quality could make a significant impact on the performance of the trained model [13]. Another barrier that limits the representational power of HGCNNs is that they only aggregate message of vertices in a localized range, while neglecting to explore information about long-range relations [11].

Several works have made their attempts to remedy these issues. DHSL [14] proposes to use the initial raw graph to update the hypergraph structure, but it fails to capture high-order relations among features. Additionally, the optimization algorithm in DHSL could be expensive cost and unable to incorporate with convolutional networks. DHGNN [15] manages to capture local and global features, but the adopted K-NN method [16] leads the graph structure to a k-uniform hypergraph and lost its flexibility. AGCN [17] proposes a spectral graph convolution network that can optimize graph adjacent matrix during training. Unfortunately, it can not be naturally extended to hypergraph spectral learning for which is based on the incidence matrix.

In a hypergraph, the incidence matrix represents its topology by recording the connections and connection strength between vertices and hyperedges. As an intuition, one could parameterize such a matrix and involve it in the end-to-end training process of the network. To this end, we propose a novel hypergraph Laplacian adaptor (HERALD), the *first* fully learnable module designed for adaptively optimizing the hypergraph structure. Specifically, HERALD takes the node features and the pre-defined hypergraph Laplacian as input and then constructs the parameterized distance matrix be-

---

★The first two authors contribute equally to this work.
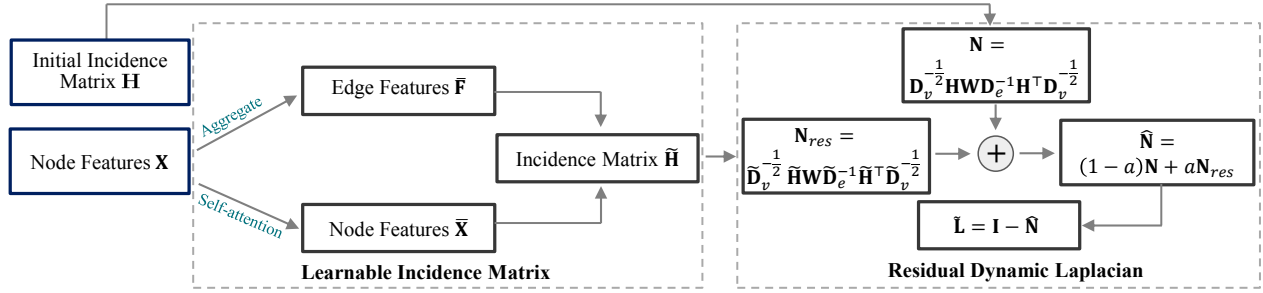†Corresponding author: Xi Xiao

Fig. 1: The schemata of proposed HERALD for hypergraph learning.

tween nodes and hyperedges, which empowers the automated updating of the hypergraph Laplacian, and thus the topology is adapted for the downstream task. Notably, HERALD employs the self-attention mechanism to model the non-local paired-nodes relation for embedding the global property of the hypergraph into the learned topology. In the experiments, to evaluate the performance of the proposed module, we have conducted experiments for hypergraph classification tasks on node-level and graph-level. Our main contributions are summarized as below:

i) We propose a generic and plug-and-play module, termed HypERgrAph Laplacian aDaptor (HERALD), for automated adapting the hypergraph topology to the specific downstream task. It is the first learnable module that can update hypergraph structure dynamically.

ii) HERALD adopts the self-attention mechanism to capture global information on the hypergraph and the parameterized distance matrix is built, which empowers the learning of the topology in an end-to-end manner.

ii) We have conducted extensive experiments on node classification and graph classification tasks, and the results show that consistent and considerable performance improvement is obtained, which verifies the effectiveness of the proposed approach.

2. PRELIMINARIES

Notation. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represents the input hypergraph with vertex set of $\mathcal{V}$ and hyperedge set of $\mathcal{E}$. $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ denotes initial vertex features. Hyperedge weights are assigned by a diagonal matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$ with each entry of $w(e)$. 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 vertex and hyperedge are defined as $d(v) = \sum_{e \in \mathcal{E}} w(e)h(v, e)$ and $\delta(e) = \sum_{v \in \mathcal{V}} h(v, e)$ which can be denoted by diagonal matrixes $\mathbf{D}_v \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ and $\mathbf{D}_e \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$, respectively. To simplify the notation, we define $\mathbf{N} := \mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$.

HGNN [10]. Zhou et al [7] define the normalized hypergraph Laplacian as follows:

$$\mathbf{L} = \mathbf{I} - \mathbf{N} \qquad (1)$$

Based on it, Feng et al [10] utilize the derivation of Defferrard et al [2] to get the spectral convolution. Specifically, i) they perform spectral decomposition of $\mathbf{L}$: $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^\top$,

where $\mathbf{U} = \{u_1, u_2, ..., u_n\}$ is the set of eigenvectors and $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ contains the corresponding eigenvalues. Using the Fourier transform, they deduce the spectral convolution of signal $\mathbf{x}$ and filter g:

$$\mathbf{g} \star \mathbf{x} = \mathbf{U}g(\Lambda)\mathbf{U}^\top\mathbf{x}, \qquad (2)$$

where $g(\Lambda) = diag(g(\lambda_1), ..., g(\lambda_n))$ is a function of the Fourier coefficients. ii) They parametrize $g(\Lambda)$ with K order polynomials $g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k$. Thus, the convolution calculation for hypergraph results in:

$$\mathbf{g} \star \mathbf{x} = \mathbf{U} \sum_{k=0}^{K-1} \theta_k \Lambda^k \mathbf{U}^\top\mathbf{x} = \sum_{k=0}^{K-1} \theta_k \mathbf{L}^k \mathbf{x}. \qquad (3)$$

where $\mathbf{L}$ is the Laplacian matrix in Eq. (1) and θ_k is the polynomials coefficients. Finally, they adopt 2-order chebyshev polynomial and define $\theta_0 = \frac{1}{2}\theta\mathbf{D}_v^{-1/2}\mathbf{H}\mathbf{D}_e^{-1}\mathbf{H}^\top\mathbf{D}_v^{-1/2}$, $\theta_1 = -\frac{1}{2}\theta$, to deduce the layers of HGCNN:

$$\mathbf{X}^{(l+1)} = \sigma(\mathbf{N}\mathbf{X}^{(l)}\mathbf{\Theta}^{(l)}) \qquad (4)$$

where σ denotes the nonlinear activation function, $\mathbf{X}^{(l)}$ is the vertex features of hypergraph at l-th layer and $\mathbf{X}^{(0)} = \mathbf{X}$.

3. HYPERGRAPH LAPLACIAN ADAPTOR

Although we can get the spectral convolution in Eq.(3), it is noticed that the convolution kernel $g_\theta(\Lambda)$ is only a *K-localized* kernel that aggregates *K*-hop nodes to the farthest per iteration, thus restricting the flexibility of kernel. Intuitively, the initial pre-defined hypergraph structure is not always the optimal one for the specific down-stream learning task. Actually, it's somehow proved that GCNs which are *K-localized* and *topology-fixed* actually simulate a polynomial filter with fixed coefficients [18, 19]. As a result, existing techniques [9, 10, 20] might neglect the modeling of non-local information and fail in obtaining high-quality hypergraph embeddings as well.

In order to improve the representaional power of HyperGraph convolutional Neural Networks (HGCNNs), we propose the HypERgrAph Laplacian aDaptor (HERALD) to dynamically optimize the hypergraph structure (a.k.a. adapt the hypergraph topology to the specific down-stream task). A straightforward way to capture the global information of graph structure is to make the filter g in Eq.(2) learnable. However, directly parametrized the filter g would lead to high

computational cost, due to the need of matrix decomposition. Hence in this paper, we choose to parametrize the Laplacian in Eq. (3) instead, to get a dynamically adaptive Laplacian $\tilde{L}$. With the dynamic Laplacian, Eq. (3) can be written as:

$$\mathbf{g} \star \mathbf{x} = \sum\nolimits_{k=0}^{K-1} \theta_k \tilde{\mathbf{L}}^k \mathbf{x}. \tag{5}$$

Next, we divide into two parts to design HERALD: Learnable Incidence Matrix and Residual Dynamic Laplacian. The framework can be seen in the Fig. 1.

3.1. Learnable Incidence Matrix

In order to learn a suitable hypergraph structure, HERALD takes the node features and the pre-defined togology to construct a parametrized incidence matrix. To be specific, given original incidence matrix $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$ and node features $\mathbf{X} = \{x_i; ..; x_{|\mathcal{V}|}\} \in \mathbb{R}^{|\mathcal{V}| \times d}$, we first get the hyperedge features $\mathbf{F} = \{f_i; ...; f_{|\mathcal{E}|}\} \in \mathbb{R}^{|\mathcal{E}| \times d}$ by averaging the features of incident vertices $f_i = \frac{1}{|e_i|} \sum_{v \in e_i} x_v$, where $|e_i|$ denotes the number of nodes in hyperedge e_i. Then we use a linear transformation to obtain the transformed hyperedge feature:

$$\bar{f}_i = \mathbf{W}_e^\top f_i, \quad i = 1, 2, ..., |\mathcal{E}| \tag{6}$$

where $\mathbf{W}_e \in \mathbb{R}^{d \times h}$ is the learnable parameter. Next, in order to enhance the representational power of the convolution kernel, HERALD adopts the self-attention mechanism [21] to encode the non-local relations between paired nodes into the updated node features $\bar{\mathbf{X}}$. That is to say, the *enhanced node features* are formulated as:

$$\bar{x}_i = \sum_{1 \le j \le |\mathcal{V}|} \alpha_{ij} \mathbf{W}_v^\top x_j, \tag{7}$$

where $\mathbf{W}_v \in \mathbb{R}^{d \times h}$ is a learnable parameter matrix for getting the same dimensions as *enhanced node features* in Eq. (6). The attention weights are calculated by:

$$\alpha_{i,j} = \frac{\exp\left((\mathbf{W}_v^\top x_i)^\top (\mathbf{W}_v^\top x_j)\right)}{\sum_{1 \le k \le |\mathcal{V}|} \exp\left((\mathbf{W}_v^\top x_i)^\top (\mathbf{W}_v^\top x_k)\right)} \tag{8}$$

With the generated node features $\bar{\mathbf{X}}\{\bar{x}_1, ..., \bar{x}_{|\mathcal{V}|}\}$ and hyperedge features $\bar{\mathbf{F}} = \{f_1, ..., f_{|\mathcal{E}|}\}$, we calculate the Hardamard power [22] of each pair of vertex and hyperedge. And then we obtain the pseudo-euclidean distance matrix of hyperedge and vertex after a linear transformation:

$$d_{ij} = \mathbf{w}_s^\top (\bar{x}_i - \bar{f}_j)^{\circ 2}, \ 1 \le i \le |\mathcal{V}|, 1 \le j \le |\mathcal{E}|, \tag{9}$$

where $\mathbf{w}_s \in \mathbb{R}^{h \times 1}$ is a learnable vector and $(\cdot)^{\circ 2}$ denotes the Hardamard power. Finally, the learnable incidence matrix $\tilde{\mathbf{H}}$ is constructed by further parameterizing the generated distance matrix with a Gaussian kernel, in which each element represents the probability that the paired node-hyperedge is connected:

$$\tilde{\mathbf{H}}_{i,j} = exp(-d_{ij}/2\sigma^2), \tag{10}$$

where σ is the hyper-parameter to control the flatness of the distance matrix. Compared with $\mathbf{H}$ that only records the incident relations between hyperedges and nodes, $\tilde{\mathbf{H}}$ contains the quantitative information of all node-edge pairs, implying the HGCNNs is able to capture the fine-grained global information via the $\tilde{\mathbf{H}}$.

3.2. Residual Dynamic Laplacian

Based on $\tilde{\mathbf{H}}$ and Eq. (1), HERALD can output the hypergraph Laplacian $\mathbf{I} - \tilde{\mathbf{N}} = \mathbf{I} - \tilde{\mathbf{D}}_v^{-1/2} \tilde{\mathbf{H}} \mathbf{W} \tilde{\mathbf{D}}_e^{-1} \tilde{\mathbf{H}}^\top \tilde{\mathbf{D}}_v^{-1/2}$ directly. However, learning the hypergraph topology from scratch may spend the expensive cost for optimization converge due to the lacking prior knowledge about a proper initialization on the parameters. To this end, we reuse the intrinsic graph structure to accelerate the training and increase the training stability. Formally, we assume that the optimal Laplacian is small shifting from the original Laplacian, in other words, the optimal $\hat{\mathbf{N}}$ is slightly shifting away from $\mathbf{N}$:

$$\hat{\mathbf{N}} = (1 - a)\mathbf{N} + a\mathbf{N}_{res}, \tag{11}$$

where a is the hyper-parameter that controls the updating strength of the topology. From this respect, the HERALD module learns the residual $\mathbf{N}_{res} = \tilde{\mathbf{D}}_v^{-1/2} \tilde{\mathbf{H}} \mathbf{W} \tilde{\mathbf{D}}_e^{-1} \tilde{\mathbf{H}}^\top \tilde{\mathbf{D}}_v^{-1/2}$ rather than $\hat{\mathbf{N}}$ and the Dynamic Laplacian is $\tilde{\mathbf{L}} = \mathbf{I} - \hat{\mathbf{N}}$. The HERALD can be inserted to per layer of HGCNNs for utilizing the topological information implied in node embeddings to obtain layer-wise hypergraph structure. Finally, the algorithm of HERALD is summarized as follows:

Algorithm 1: HERALD module

Input: **Node embeddings produced from** l-th-**layer of HGCNNs**: $\mathbf{X}^{(l)} = \{x_i^{(l)}\}_{i=1}^{|\mathcal{V}|}$;
Initial Laplacian matrix $\mathbf{L}$ and $\mathbf{N} = \mathbf{I} - \mathbf{L}$.
1: $\tilde{\mathbf{H}}^{(l)} \leftarrow$ Eq.(6-10) // Learnable Incidence Matrix
2: $\mathbf{N}_{res}^{(l)} = (\tilde{\mathbf{D}}_v^{(l)})^{-1/2} \tilde{\mathbf{H}}^{(l)} \mathbf{W} (\tilde{\mathbf{D}}_e^{(l)})^{-1} (\tilde{\mathbf{H}}^{(l)})^\top (\tilde{\mathbf{D}}_v^{(l)})^{-1/2}$
3: $\hat{\mathbf{N}}^{(l)} = (1 - a)\mathbf{N} + a\mathbf{N}_{res}$
4: $\tilde{\mathbf{L}}^{(l)} = \mathbf{I} - \hat{\mathbf{N}}^{(l)}$ // Residual Dynamic Laplacian
Output: $\tilde{\mathbf{L}}^{(l)}$

To sum up, the **learning complexity** of HERALD module is $\mathcal{O}(d_i h)$ (d_i is the dimension of the input features at layer i) with introduced parameters $\{\mathbf{W}_v, \mathbf{W}_e, \mathbf{w}_s\}$, independent of the input hypergraph size and node degrees.

4. EXPERIMENTS

In the experiments, we evaluate the proposed HERALD on node classification and graph classification tasks, and we select HGNN [10] to work as the evaluation backbone. The vanilla GCN [1] also acts as a baseline. The Adam [23] optimizer is used to train the backbone with HERALD for 1000 epochs with a learning rate of 0.01 and early stopping with a patience of 100 epochs on Tesla P40.

4.1. Hypernode Classification

Datasets. This task is semi-supervised node classification. we employ two hypergraph datasets: co-citation relationship and co-authorship of Cora [4], which are released by Yadat et al. [9]. We adopt the datasets and train-test splits (10

Table 1: The results on the task of hypergraph classification. We report the average test accuracy and its standard deviation under the 10-fold cross-validation.

Datasets	MUTAG	PTC	IMDB-B	PROTEINS	NCI1	COLLAB
GCN [1]	68.60 ± 5.6	65.41 ± 3.7	53.00 ± 1.9	68.29 ± 3.7	58.08 ± 1.1	52.69 ± 0.5
HGNN [10]	69.12 ± 6.2	66.56 ± 4.7	55.20 ± 3.7	68.38 ± 3.8	58.32 ± 1.3	55.61 ± 2.5
HGNN + HERALD	$\mathbf{71.23 \pm 9.0}$	$\mathbf{67.75 \pm 5.9}$	$\mathbf{58.20 \pm 5.5}$	$\mathbf{68.64 \pm 3.4}$	$\mathbf{58.37 \pm 1.4}$	$\mathbf{55.74 \pm 2.3}$

Table 2: The results on the task of node classification. We report the average test accuracy and its standard deviation of test accuracy under 10 runs with different random seeds.

Method	Cora (co-citation)	Cora (co-authorship)
HGNN [10]	48.23 ± 0.2	69.21 ± 0.3
HGNN + HERALD	$\mathbf{57.31 \pm 0.2}$	70.05 ± 0.3
HGNN + FastHERALD	57.27 ± 0.3	$\mathbf{70.16 \pm 0.4}$

splits) as provided in their publicly available implementa- tion (https://github.com/malllabiisc/HyperGCN). Specifically, the co-authorship data consists of a collection of the papers with their authors and the co-citation data consists of a collection of the papers and their citation relationship.

Experimental Setup. We use a 3-layer HGNN as a baseline. We perform a random search on the hyper-parameters and report the case giving the best accuracy on the validation set. To evaluate the proposed module, we add the HERALD module to the latter two layers in HGNN. That is to say, in the l-th layer (where $2 \leq l \leq 3$), HERALD generates $\hat{\mathbf{N}}^{(l)}$, and the feature updating function is given by $\mathbf{X}^{(l+1)} = \sigma(\hat{\mathbf{N}}^{(l)}\mathbf{X}^{(l)}\mathbf{\Theta}^{(l)})$. We set $a = 1 - 0.9 * (\cos(\pi(l-1)/10) + 1)/2$ in Eq. (11) for gradually increase the updating strength of the task-specific adapted topology. We also add a loss regularizer of $||\mathbf{N} - \mathbf{N}_{res}||_2$ to make the training more stable, of which the loss weight is fixed to 0.1.

Fast-HERALD. For designing a more cost-friendly method, we also propose a variant of the usage of HERALD, named Fast-HERALD. It constructs $\hat{\mathbf{N}}$ at the beginning HGNN layer and reuses it in the process of feature updating for all the rest layers. Since $\hat{\mathbf{N}}$ is shared on each layer, it can reduce the number of parameters and increase the training speed.

Results and Disscussion. The results of experiments for hypernode classification are given in Table 2. It is observed that HERALD consistently improves the testing accuracy for all the cases. It gains 0.84% improvement on the Co-authorship Cora dataset while achieving a remarkable 9.08% increase on the Co-citation Cora dataset. We also notice the FastHERALD gets the best result on co-authorship Cora. The results show that our proposed module can significantly improve the performance of the hypergraph convolutional network by adapting the topology to the downstream task.

4.2. Hypergraph Classification

Datasets. The datasets we use include: MUTAG, PTC, NCI1, PROTEINS, IMDB-BINARY, COLLAB [3], which are released by Xu *et al.* [24] (https://github.com/weihua916/powerful-gnns). We use the same datasets and data splits of Xu *et al.* [24]. Note that all those datasets are simple graphs,

and we employ the method proposed by Feng *et al.* [10] to generate hypergraph structure, i.e. each node is selected as the centroid and its connected nodes form a hyperedge including the centroid itself.

Experimental Setup. We also use HGNN as the evaluation backbone and employ the same hyper-parameter search process as in the previous experiment. And we conduct controlled experiments with the difference of with and without the plugging of the HERALD module. The hyper-parameters a are used the same settings as the illustration stated before. For obtaining the hypergraph embedding, we add a summation operator as the permutation invariant layer at the end of the backbone to readout the node embeddings.

Results and Disscussion. The results of experiments for hypergraph classification are shown in Table 1. Comparing with the results of vanilla GCN and HGNN, it can be observed that the performance of HGNN is better than that of GCN, which demonstrates the meaning of using hypergraph to work as a more powerful tool for modeling complex irregular relationships. One can also see that the proposed HERALD module consistently improves the model performance for all the cases and gains 1.12% test accuracy improvement on average, which further verifies the effectiveness and generalization of the approach.

5. CONCLUSION

We have presented a generic plug-and-play module of HypERgrAph Laplacian aDaptor (HERALD) for improving the representational power of HGCNNs. The module is tailored to design for constructing task-aware hypergraph topology. To this end, HERALD generates the parameterized hypergraph Laplacian and involves it in the end-to-end training process of HGCNNs. The experiments have shown our method gained remarkable performance on both hypernode and hypergraph tasks, which verifies the effectiveness of the method.

6. ACKNOWLEDGMENTS

References

[1] Thomas N Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the ICLR*, 2017.

[2] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[3] Pinar Yanardag and SVN Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1365–1374.

[4] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[5] Xiang Yue, Zhen Wang, Jingong Huang, Srinivasan Parthasarathy, Soheil Moosavinasab, Yungui Huang, Simon M Lin, Wen Zhang, Ping Zhang, and Huan Sun, "Graph embedding on biomedical networks: methods, applications and evaluations," *Bioinformatics*, vol. 36, no. 4, pp. 1241–1251, 2020.

[6] Elena V Konstantinova and Vladimir A Skorobogatov, "Application of hypergraph theory in chemistry," *Discrete Mathematics*, vol. 235, no. 1-3, pp. 365–383, 2001.

[7] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," *Advances in neural information processing systems*, vol. 19, pp. 1601–1608, 2006.

[8] Ke Tu, Peng Cui, Xiao Wang, Fei Wang, and Wenwu Zhu, "Structural deep embedding for hyper-networks," in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2018.

[9] 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*, 2019, pp. 1511–1522.

[10] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao, "Hypergraph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 3558–3565.

[11] 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.

[12] Jing Huang and Jie Yang, "Unignn: a unified framework for graph and hypergraph neural networks," in *IJCAI-21*, 2021.

[13] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang, "Deep graph structure learning for robust representations: A survey," *arXiv preprint arXiv:2103.03036*, 2021.

[14] Zizhao Zhang, Haojie Lin, Yue Gao, and KLISS BNRist, "Dynamic hypergraph structure learning.," in *IJCAI*, 2018, pp. 3162–3169.

[15] Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao, "Dynamic hypergraph neural networks.," in *IJCAI*, 2019, pp. 2635–2641.

[16] Naomi S Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[17] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang, "Adaptive graph convolutional neural networks," *arXiv preprint arXiv:1801.03226*, 2018.

[18] Qimai Li, Zhichao Han, and Xiao-Ming Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," *arXiv preprint arXiv:1801.07606*, 2018.

[19] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li, "Simple and deep graph convolutional networks," *arXiv preprint arXiv:2007.02133*, 2020.

[20] Yihe Dong, Will Sawin, and Yoshua Bengio, "Hnhn: Hypergraph networks with hyperedge neurons," *arXiv preprint arXiv:2006.12278*, 2020.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[22] Ruochi Zhang, Yuesong Zou, and Jian Ma, "Hypersagnn: a self-attention based graph neural network for hypergraphs," *arXiv preprint arXiv:1911.02613*, 2019.

[23] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka, "How powerful are graph neural networks?," in *ICLR*, 2018.