ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230379

LoSS: Local Structural Separation Hypergraph Convolutional Neural Network

Bingde Hu[acd], Yang Gao[a], Zunlei Feng[bcd;*], Mingli Song[acd], Xinyu Wang[a] and Ying Li[e]

[a]College of Computer Science and Technology, Zhejiang University
[b]College of Software Technology, Zhejiang University
[c]ZJU-Bangsun Joint Research Center
[d]Shanghai Institute for Advanced Study of Zhejiang University
[e]Bangsun Technology

Abstract. Graph classification is a classic problem with practical applications in many real-life scenes. Existing graph neural networks, including GCN, GAT, and GIN, are proposed to extract useful features from complex graph structures. However, most existing methods' feature extraction and aggregation inevitably mix the useful and redundant features, which will disturb the final classification performance. In this paper, to handle the above drawback, we put forward the Local Structural Separation Hypergraph Convolutional Neural Network (LoSS) based on two discoveries: most graph classification tasks only focus on a few groups of adjacent nodes, and different categories have their specific high response bits in graph embeddings. In LoSS, we first decouple the original graph into different hypergraphs and aggregate the features in each substructure, which aims to find useful features for the final classification. Next, the low-correlation feature suppression strategy is devised to suppress the irrelevant node-level and bit-level features in the forward inference process, effectively reducing the disturbance of redundant features. Experiments on five datasets show that the proposed LoSS can effectively locate and aggregate useful hypergraph features and achieve SOTA performance compared with existing methods.

1 Introduction

Graph classification is an important research task used for solving various real-life classification problems, like identifying the properties of different proteins, searching for molecules with specific functions, discovering hotspots of social networks, etc. Graphs used in the above scenes usually involve more than hundreds/ thousands of nodes and complex connections, which contain useful and redundant features for the classification task.

Many researchers propose a series of works to fully explore the useful features in the graphs, including GAT[20], GCN[22], GIN[24], etc. GCN adopts the graph Laplacian to capture the structural information and activation function to filter out redundant features. GAT uses the attention mechanism to pay more attention to potentially useful features for the specific task. GIN tries to extract useful features with the neighborhood aggregation technique.

However, most existing works directly aggregate the messages from first-order neighbors of each node. The simple first-order information aggregation mechanism inevitably mixes the useful features and redundant features, which is harmful to the classification result. What's more, two discoveries concluded by preliminary analysis also show that the devised strategies of existing frameworks are not entirely harmonious with the underlying calculation mechanism of the graph classification model. As shown in Figure 1(a), we find out that *most graph classification tasks only focus on a few groups of adjacent nodes*. Another discovery is that *different categories have their specific bits with high responses in the graph embedding (the summed 1-d feature vector before the full connection classifier layer)*. Please refer to Section 3.1 for more details about the above two discoveries.

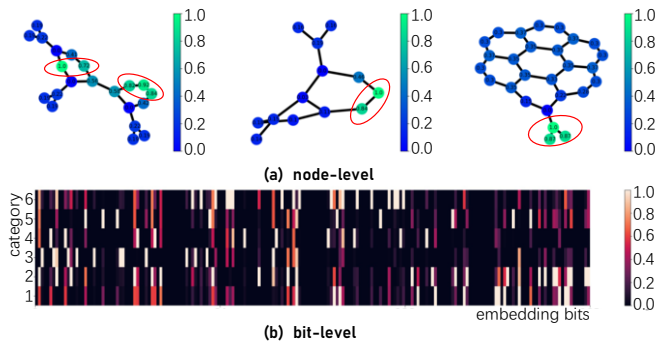

Figure 1. The illustration of two discoveries is relevant to node-level features and bit-level features. a) Graph classification tasks focus on a few groups of adjacent nodes. b) Each category has specific bits with high responses in the graph embedding (the summed 1-d feature vector before the full connection classifier layer).

The incompatibility between the above two discoveries and existing feature extraction strategies raises two questions: a) *How to well and truly find important features from the complex graph structure?* b) *How to effectively aggregate and calculate those important features for the final classification?*

In this paper, we put forward a novel framework for the graph classification task, termed Local Structural Separation Hypergraph Convolutional Neural Network (LoSS), in response to the above two questions. In order to find the potentially useful features from the complex graph structure, we devise a feature decoupling and group aggregation strategy based on the first discovery (Figure 1(a)). Firstly, the original graph is decoupled into multi channels with local structural characteristics (Probabilistic random walk clustering). For

* Corresponding Author. Email: zunleifeng@zju.edu.cn

each channel, graph nodes are separated into different groups based on corresponding random walks, which will be modeled by hypergraphs. Then, the features of nodes in each group are aggregated in the forward inference process and finally concatenated together for classification.

Furthermore, with the decoupled and aggregated features, we propose a low-correlation suppression strategy based on two discoveries (Figure 1). The low-correlation feature suppression strategy is devised to reduce the disturbance of redundant features for the model decision-making by suppressing the redundant hypergraph features generated in the forward inference process. With the above two strategies, useful graph features can be founded in the forward inference process through decoupling, aggregating, and suppressing operations under the guidance of the supervision loss function, which can reduce the impact of redundant features effectively. Experiment results show that the proposed LoSS can effectively locate and aggregate useful hypergraph features for final classification, which achieves SOTA performance compared with existing methods.

Our contribution is, therefore, the proposed LoSS based on two discoveries. The local structure based decoupling and aggregation strategy is devised to decouple the original graph into different hypergraphs and aggregate the features in each substructure, which can fundamentally find useful features for the final classification. Moreover, the low-correlation feature suppression strategy is devised to suppress the redundant features of hypergraphs in the forward inference process, which can effectively reduce the disturbance of redundant features. Meanwhile, the proposed LoSS achieves SOTA performance compared with existing methods.

2 Related Work

Graph convolutional network. Inspired by convolutional neural networks [9], graph convolutional neural networks are proposed, which are applied to non-Euclidean data structures (graph). Bruna *et al.*[2] first use graph Laplacian eigenvectors as graph Fourier basis and propose the spectral graph convolution operation. To improve the efficiency of spectral graph convolution operation,[22] and [3] are proposed, which use Chebyshev polynomials instead of the Laplacian eigendecomposition. To generate Low-dimensional embeddings of unseen nodes, the flexible framework GraphSAGE[6] is designed, which learns a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. Xu *et al.*[24] analyze the expressiveness of graph neural networks and introduce a simple but powerful architecture Graph Isomorphism Network. Vincent *et al.*[21] use the Fused Gromov-Wasserstein distance to enhance the structure discrimination ability of GNN.

Hypergraph Convolutional Network. To explore hypergraphs, some hypergraph neural networks are proposed. Inspired by graph convolution neural networks, the first hypergraph convolution neural network HGNN[4] is proposed, which encodes high-order data correlation in a hypergraph structure. Dynamic hypergraph structure learning DHSL[31] is the first dynamic hypergraph structure learning method, which simultaneously optimizes the label projection matrix and the hypergraph structure itself. However, DHSL only updates the hypergraph structure on initial feature embedding. To tackle this issue, DHGNN[7] is designed to update the hypergraph structure in each model layer. To accelerate the calculation of hypergraph neural networks, HyperGCN[25] expands hypergraphs to pair-wise graphs and samples the relations.

Attribution Method. Backpropagation-based methods are one of the most important attribution methods[30]. To meet two fundamental axioms-Sensitivity and Implementation Invariance, Sundararajan *et al.*[19] design the attribution method Integrated Gradient, which attributes the prediction of a deep network to its input features. Shrikumar *et al.*[16] propose DeepLIFT to compare the activation of each neuron to its 'reference activation' and assigns contribution scores according to the results of backpropagating the contributions. Feng *et al.*[5] introduce the aggregate gradient strategy to effectively diagnose and optimize CNN classifiers. Sebastian *et al.*[1] propose the Layer-Wise Relevance Propagation method to propagate the prediction backward in the neural network, using predefined local propagation rules and get a relevance score for each neural unit. Based on LRP method and gradient method, Song *et al.*[17] calculate the attribution map to explore the transferability between heterogeneous tasks. Schnake *et al.*[13] migrate the LRP method to the graph neural network and explain the decision-making mechanism of the graph neural network. Jing *et al.*[8] introduce a topological attribution map to highlight the structural saliency in a graph and amalgamate model knowledge. Yang*et al.*[28][27] use attribution methods to study the factorization and reassembly of model knowledge. Liu*et al.*[10][11] use attribution methods to study the condensation of datasets.

3 Methodology

In this section, we first introduce two discoveries about the latent forward inference mechanism of GNN (Section 3.1). Then, based on the first discovery, a probabilistic random walk clustering based decoupling and aggregation strategy is devised for extracting the potential useful group features for the graph classification task (Section 3.2). Next, low-correlation feature suppression is proposed to reduce the disturbance of redundant node-level and bit-level features in the forward inference process (Section 3.3).

3.1 Discoveries and Analysis

Taking the GCN model as an example, we show the statistical results of two discoveries in Figure 1. A detailed analysis will be given in the following section.

Discovery 1: *Most graph classification tasks only focus on a few groups of adjacent nodes.*

Analysis: For the graph classification task, a model usually is used to get the node features via the graph. We denote the final node features obtained from the model as $\{x_1, x_2, x_3, ..., x_t, ..., x_T\}$, and the graph embedding can be expressed as:

$$f = \sum \{x_1, x_2, x_3, ..., x_t, ..., x_T\}, \tag{1}$$

where f is the sum of the nodes features belonging to the graph, T denotes the node number, and $\sum$ denotes the element-wise adding operation. The graph embedding f will be used to distinguish the corresponding graph. Thus, the correlation of each node can be considered as the correlation of the node features to the graph embedding. The correlation of node t is calculated as follows:

$$r_t = \sum_i \frac{x_t[i]}{f[i]}, \tag{2}$$

where $x_t[i]$ and $f[i]$ denote the i-th value in feature vector x_t and $f[i]$, respectively. We normalize the nodes correlations and show the

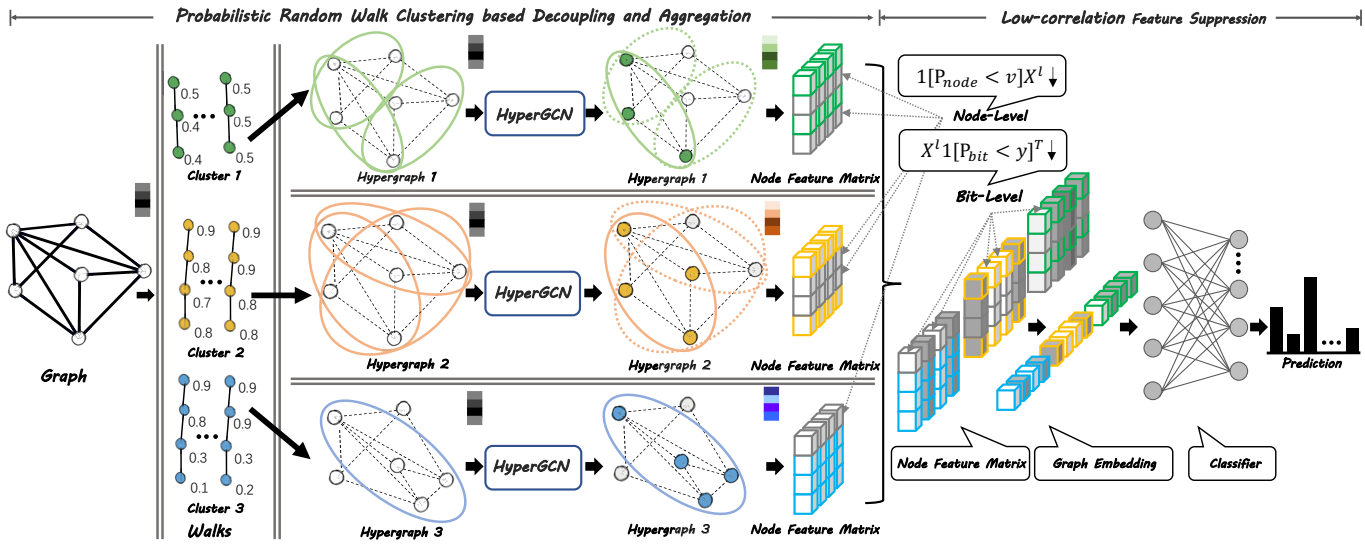

Figure 2. The framework of LoSS, which extracts and aggregates useful features for graph classification and suppresses low-correlation features from node-level and bit-level. The graph is firstly decoupled into different hypergraphs by probabilistic random walk clustering. Then, HyperGCN is introduced to aggregate the features of each hyperedge (a group of nodes denoted by green/orange/blue ellipses), which can extract the potential useful node group features for the final classification. Furthermore, the low-correlation feature suppression strategy is devised to suppress redundant bit-level and node-level features (denoted by the vertical and horizontal white cube, respectively) in the node feature matrix, which can effectively reduce the disturbance of redundant bit-level and node-level features.

results of high predict confidence graph samples in Figure 1(a). From the results, we discover that only some useful nodes features play significant role in the graph classification task (i.e., have high correlations). Based on the discovery, we put forward the probabilistic random walk clustering based decoupling and aggregation strategy for extracting the potentially useful features, and node-level low-correlation feature suppression to suppress redundant node features.

Discovery 2: *Different categories have their specific bits with high responses in the graph embedding.*

Analysis: In this section, we analyze the correlation between the model's prediction and the bit-level features of graph embedding. With the summed graph embedding f as input, the prediction of the full connection classifier can be denoted as $p = f\mathbf{W}^c$, where $\mathbf{W}^c$ denotes the weight matrix of the classifier. Then, we adopt the LRP-$\alpha\beta$ rule[1] to calculate the contribution of each graph embedding bit to the target category, which is given as follows:

$$c^k[j] = \begin{cases} \alpha \frac{(f[j]\mathbf{W}^c[j,k])}{\sum_j (f[j]\mathbf{W}^c[j,k])} & \text{if } f[j]\mathbf{W}^c[j,k] > 0 \\ 0 & \text{if } f[j]\mathbf{W}^c[j,k] = 0 \\ \beta \frac{(f[j]\mathbf{W}^c[j,k])}{\sum_j (f[j]\mathbf{W}^c[j,k])} & \text{if } f[j]\mathbf{W}^c[j,k] < 0 \end{cases}, \quad (3)$$

where $\alpha+\beta = 1$, $\mathbf{W}^c$ is the weight matrix of the classifier, $\mathbf{W}^c[j,k]$ are the value of row j and column k in matrix $\mathbf{W}^c$,$c^k[j]$ is the correlation of graph embedding bit j to class k.

With the above correlation attribution technique, we select 50 high confidence samples to analyze the correlation of each graph embedding bit to the target category. Figure 1(b) shows the average bit correlation for each category, which demonstrates that samples of the same category have specific and fixed bits with high correlation. Thus, we propose the low-correlation feature suppression strategy to reduce the disturbance of redundant features.

3.2 *Probabilistic Random Walk Clustering based Decoupling and Aggregation*

In this section, we will introduce the technique used to decouple the original graph into multi channels with Probabilistic Random Walk Clustering. Firstly, we generate the random walks for each node, and these walks are collected in a matrix $\mathbf{T}$. The row vector of $\mathbf{T}$ represents a random walk path, which consists of nodes' index. Then, the frequency of each node appearing in each column of matrix T is counted, and the probability of each node occurrence is calculated. The nodes' probabilities are used to replace the nodes' indexes in $\mathbf{T}$. The walks are clustered (k-means) based on the probability representation of each path in $\mathbf{T}$.

The probability of a node being accessed reflects its local structural characteristics, so the clustering results are divided based on the local structural characteristics of the path. The paths assigned to the same cluster have similar probability distributions. The different clusters will be constructed into different hypergraphs separately (a path is a hyperedge). The hypergraphs constructed by different clusters describe different structural characteristics of the graph, which will be processed by independent channels, as shown in Figure 2.

The hypergraph is a generalization of the pair-wise graph in which an edge can join any number of nodes. A hypergraph can be represented as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is a set of nodes, $\mathcal{E}$ is a set of hyperedges. The node relations in the hypergraph can be represented by an incidence matrix $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$, which is defined as:

$$\mathbf{H}(v,e) = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{otherwise} \end{cases}, \quad (4)$$

where $e \in \mathcal{E}, v \in \mathcal{V}$.

Different probabilistic random walk clusters generate different hypergraphs, as shown in Figure 2. The incidence matrix of hypergraph, which is generated by probabilistic random walk cluster g_i, is rep-

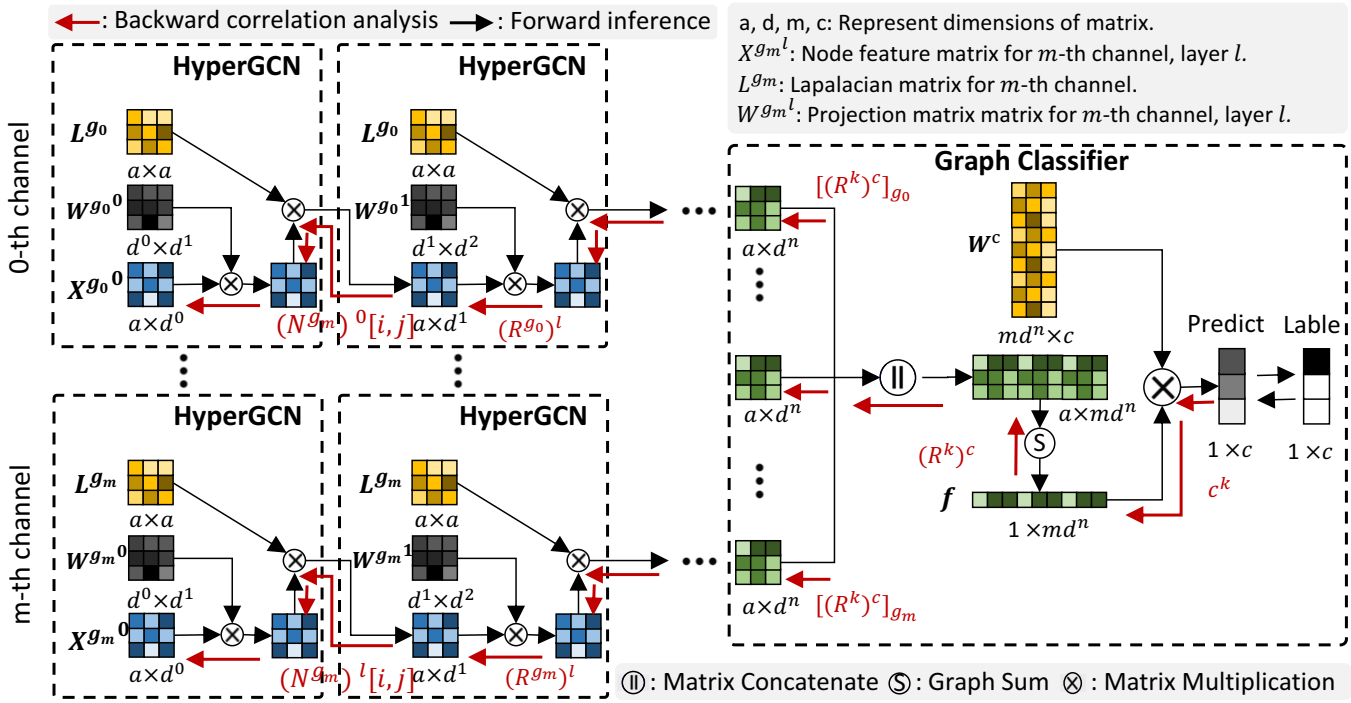

Figure 3. An illustration of correlation analysis for forward inference and backward correlation analysis. The black arrows represent the forward inference process of the model, and the red arrows represent the backward correlation analysis process. In the backward correlation analysis process, we attribute the prediction results to the input node feature matrix and analyze useful features from the node-level and bit-level.

resented by $\mathbf{H}^{\mathbf{g_i}}$. The Laplacian matrix of HGNN [4] is adopted to aggregate the features of the hypergraph $\mathbf{H}^{\mathbf{g_i}}$, which is termed as HyperGCN module in this paper. The aggregation process is formulated as follows:

$$(\mathbf{X}^{g_i})^l = (\mathbf{D}_v^{g_i})^{-1}\mathbf{H}^{g_i}(\mathbf{D}_e^{g_i})^{-1}(\mathbf{H}^{g_i})^T(\mathbf{X}^{g_i})^{l-1}(\mathbf{W}^{g_i})^l, \quad (5)$$

where $(\mathbf{X}^{g_i})^0$ is initial node features matrix, l is the number of layers, $\mathbf{H}^{g_i}$ is the hypergraph incidence matrix constructed by probabilistic random walk cluster g_i, and the adopted hypergraph Laplacian is $(\mathbf{D}_v^{g_i})^{-1}\mathbf{H}^{g_i}(\mathbf{D}_e^{g_i})^{-1}(\mathbf{H}^{g_i})^T$. Diagonal matrices $\mathbf{D}_e^{g_i} = \boldsymbol{diag}(\mathbb{1}\mathbf{H}^{g_i})$ and $\mathbf{D}_v^{g_i} = \boldsymbol{diag}(\mathbf{H}^{g_i}\mathbb{1})$ represent hyperedge degree matrix and hypernode degree matrix respectively, where $\mathbb{1}$ is all-one vector.

The Laplacian matrix determines the propagation mode of node message in the hyperedges, and different probabilistic random walk clusters make the hyperedges have different perceptual fields. Thus, LoSS is enabled to capture multi-scale structural information and reduce the coupling of graph features.

The hypergraphs generated by different probabilistic random walk clusters for the original graph are processed by separate channels. Therefore, after the aggregation process, different hypergraphs result in different features, which will be concatenated together as follows:

$$\mathbf{X} = ||_{i=0}^{K-1}\mathbf{X}^{g_i}, \quad (6)$$

where K is the number of probabilistic random walk clusters, and $||$ is concatenate operation. Concatenation operation maintains the independent of the features in the downstream tasks. Then, the graph embedding f is obtained through summing the concatenated node feature matrix in the vertical dimension as follows:

$$f = \boldsymbol{graph}_{\boldsymbol{sum}}\mathbf{X}, \quad (7)$$

which converts the two-dimension matrix into one-dimension vector. The obtained graph embedding f is the input of the full connection classifier layer.

3.3 *Low-correlation Feature Suppression*

The above two discoveries show that there are useful and redundant features in the node-level and bit-level. Therefore, we devise the low-correlation feature suppression strategy to reduce the disturbance of redundant node-level and bit-level features.

In this section, we first analyze the correlation between the node features and the prediction results. Then, based on the correlation results, the node-level and bit-level suppression strategies are used to reduce the disturbance of node-level and bit-level redundant features. To make the calculation process easier to understand, we draw the calculation process of correlation analysis in Figure 3.

Correlation analysis. Correlation analysis can be divided into two processes: Classifier analysis and HyperGCN analysis. In the process of Classifier analysis, firstly, we need to calculate the correlation of each graph embedding bit to the final classification result. And then, the correlation will be allocated to the graph nodes because the graph embedding is summed by graph node features.

The calculation process of the correlation of each graph embedding bit to the final classification is the same as Eqn. (3), which is used in Section 3.1. We also use c^k to represent the contribution of graph embedding, where $c^k[j]$ represents the contribution of embedding bit j to the class k. Then, we allocate the contribution of graph

embedding to the graph nodes. The calculation process can be formulated as:

$$(\mathbf{R}^k)^c[i,j] = \mathbf{X}[i,j]\ \frac{1}{f[j]}\ c^k[j], \tag{8}$$

where $\mathbf{X}$ is the node feature matrix, and $(\mathbf{R}^k)^c[i,j]$ represents the contribution of the j-th bit of node i's feature. After that, we can get the decision-making correlation coefficient for each node and bit in the classifier.

Based on the result of the Classifier analysis, we further analyze the impact of HyperGCN's input node features on the final classification result. In the process of HyperGCN analysis, it can also be divided into two parts: Propagation analysis and Projection matrix analysis. When using Laplacian matrix for message propagation, a node will influence the other nodes belonging to the same hyperedges. Thus, the Propagation analysis is carried out to analyze the influence of a node's feature on the other nodes' features during the propagation process. According to the reverse propagation process of Eqn. (5), we get the feature correlation tracing formula in the process of propagation as follows:

$$\begin{aligned}(\mathbf{R}^{g_i})^l = ((\mathbf{X}^{g_i})^{l-1}(\mathbf{W}^{g_i})^l) * (\mathbf{H}^{g_i}(\mathbf{D_e}^{g_i})^{-1}(\mathbf{H}^{g_i})^T \\ (\mathbf{D_v}^{g_i})^{-1})(\frac{1}{(\mathbf{X}^{g_i})^l} * [(\mathbf{R}^k)^c]_{g_i})),\end{aligned} \tag{9}$$

where $*$ is hadamard product, the elements in matrix $\frac{1}{(\mathbf{X}^{g_i})^l}$ is the reciprocal of the elements in matrix $(\mathbf{X}^{g_i})^l$, and $[\,]_{g_i}$ means taking the columns corresponding to grphlet g_i. Eqn. (9) calculates the influence of the feature matrix $(\mathbf{X}^{g_i})^{l-1}(\mathbf{W}^{g_i})^l$ to feature matrix $(\mathbf{X}^{g_i})^l$, through hypergraph Laplacian. For ease of understanding, the Eqn. (9) shows the correlation distribution of the penultimate layer (i.e., The layer before the classification layer). If the upper layer is HyperGCN, we can just replace $[(\mathbf{R}^k)^c]_{g_i})$ with $(N^{g_i})^{l+1}$ to calculate the decision-making correlation.

Since there is a projection matrix in a HyperGCN cell, we need to conduct a Projection matrix analysis. The Projection matrix analysis is similar to the process of Classifier analysis. We also adopt the LRP-$\alpha\beta$ rule[1]. It is formulated as:

$$\begin{aligned}(\mathbf{N}^{g_i})^l[i,j] = \sum_k (\alpha \frac{(\mathbf{X}^{l-1}[i,j](\mathbf{W}^{g_i})^l[j,k])^+}{\sum_j (\mathbf{X}^{l-1}[i,j](\mathbf{W}^{g_i})^l[j,k])^+} \\ -\beta \frac{(\mathbf{X}^{l-1}[i,j](\mathbf{W}^{g_i})^l[j,k])^-}{\sum_j (\mathbf{X}^{l-1}[i,j])(\mathbf{W}^{g_i})^l[j,k])^-})(\mathbf{R}^{g_i})^l[i,j],\end{aligned} \tag{10}$$

where $(\mathbf{N}^{g_i})^l[i,j]$ represents the contribution of the input feature bit j of node i in channel g_i layer l. $()^+$ and $()^-$ means to get the positive and negative part, respectively.

After calculating the decision-making correlation matrix $(\mathbf{N}^{g_i})^l$, we use it to analyze the feature importance from the node-level and bit-level. In the node-level, we first normalize $(\mathbf{N}^{g_i})^l$ by column and then add it column by column to obtain the indicator vector of node correlation. In the bit-level, we normalize $(\mathbf{N}^{g_i})^l$ by row and add it row by row so as to get the indicator vector of bit correlation. The calculation process can be formulated as:

$$p_{node}[i] = \sum_j \mathbf{M}_n[i,j], \mathbf{M}_n = \boldsymbol{norm_{col}}(\mathbf{N}^{g_i})^l, \tag{11}$$

$$p_{bit}[i] = \sum_j \mathbf{M}_b[i,j], \mathbf{M}_b = \boldsymbol{norm_{row}}(\mathbf{N}^{g_i})^l, \tag{12}$$

p_{bit} and p_{node} are two indicator vectors, which will be used to select the low decision-making correlation bits and nodes.

Feature suppression. In order to suppress the redundant features, we introduce two loss functions in this section. With the indicator vectors p_{bit} and p_{node}, we can get the feature suppression loss $\mathcal{L}_{node}$ and $\mathcal{L}_{bit}$ as follows:

$$\mathcal{L}_{node} = \mathbf{1}[p_{node} < v]X^{l-1}, \tag{13}$$

$$\mathcal{L}_{bit} = X^{l-1}\mathbf{1}[p_{bit} < y]^T, \tag{14}$$

where $\mathbf{1}[p_{node} < v]$ is a binary row vector, in which the element equals 1 if the element of p_{node} less than the threshold value v; otherwise equals 0, and $\mathbf{1}[p_{bit} < y]$ also follows this rule. At the beginning of training, we use the task loss $\mathcal{L}_{task}$ to train several epochs. And then, we get the high confidence training samples to evaluate the bit and node level correlation so that the $\mathcal{L}_{node}$ and $\mathcal{L}_{bit}$ are obtained. We introduce the $\mathcal{L}_{node}$ and $\mathcal{L}_{bit}$ and continue to train the model. The overall loss is denoted as:

$$\mathcal{L} = \mathcal{L}_{task} + \mathcal{L}_{node} + \mathcal{L}_{bit}, \tag{15}$$

The two constraints can reduce the impact of useless features, which could also be appended to the task loss separately.

4 Experiments

4.1 Datasets

Five datasets are selected to verify the performance of the proposed method LoSS. The first one is a synthetic dataset[29], which is generated from a fixed number of predefined graphs like Turán graph, house-x graph, and balanced-tree graph. The half of predefined factor graphs will be selected randomly and merged as a training sample. It's a multi-label graph classification dataset. The other four datasets are widely used multi-class graph classification datasets: IMDBBINARY, MUTAG, PTC, and PROTEINS[26].

4.2 Baselines

We compare LoSS with ten typical graph network architectures, which are listed as follows: WL[14], GK[15], DGK[26], GCN[22], GAT[20], GIN[24], DisenGCN[12], and FactorGCN[29]. WL, DGK, GK are three graph kernel graph kernel based methods. GCN, GAT, GIN, DisenGCN, and FactorGCN are graph neural network methods. HRN[23] and GAT-CAL[18] are two new method.

4.3 Experiment setting

The computational resources used are NVIDIA A40 (48G), Xeon 6342R@2.80GHz, and 256GB main memory. We implement our method in PyTorch. The proposed model is optimized by Adam optimizer, using the learning rate of 0.01, the number of training epochs is set to 500, the weight decay is set to $5e^{-5}$. ReLU is selected as activation function. The dimension of the hidden feature is set to 64. The number of hidden layers is set to four for all models. The length of random walk is set to 5, the number of clusters is also set to 5.

4.4 Quantitative Comparison Results

In this section, the results of LoSS for graph classification on the aforementioned five datasets are evaluated. The accuracy and standard deviations are reported in Table 1, where the best results are shown in bold and 10-fold cross-validation is conducted. The LoSS row in Table 1 shows the results of our proposed model without the

Methods \ Datasets	SYNTHETIC	MUTAG	PTC	PROTEINS	IMDBBINARY
WL[14]	-	0.821 ± 0.004	0.570 ± 0.020	0.729 ± 0.006	-
GK[15]	-	0.835 ± 0.006	0.597 ± 0.003	0.717 ± 0.006	0.659 ± 0.010
DGK[26]	-	0.874 ± 0.027	0.601 ± 0.026	0.757 ± 0.005	0.670 ± 0.006
GCN[22]	0.838 ± 0.001	0.845 ± 0.064	0.628 ± 0.072	0.705 ± 0.056	0.736 ± 0.025
GAT[20]	0.852 ± 0.002	0.802 ± 0.065	0.610 ± 0.046	0.759 ± 0.055	0.728 ± 0.047
GIN[24]	0.806 ± 0.005	0.894 ± 0.056	0.646 ± 0.070	0.762 ± 0.028	0.751 ± 0.051
DisenGCN[12]	0.810 ± 0.002	0.781 ± 0.077	0.593 ± 0.031	0.742 ± 0.037	0.739 ± 0.037
FactorGCN[29]	0.814 ± 0.004	0.901 ± 0.086	0.641 ± 0.041	0.764 ± 0.030	0.749 ± 0.021
HRN[23]	0.857 ± 0.004	0.904 ± 0.089	0.661 ± 0.064	0.769 ± 0.012	0.773 ± 0.059
GAT-CAL[18]	0.861 ± 0.003	0.901 ± 0.067	0.654 ± 0.035	0.768 ± 0.032	0.724 ± 0.052
LoSS (ours)	0.864 ± 0.001	0.894 ± 0.053	0.672 ± 0.036	0.775 ± 0.026	0.784 ± 0.022
+Node-Level Suppression	0.865 ± 0.002	$\mathbf{0.909 \pm 0.059}$	0.683 ± 0.036	$\mathbf{0.781 \pm 0.027}$	0.788 ± 0.022
+Bit-Level Suppression	$\mathbf{0.869 \pm 0.001}$	0.894 ± 0.053	$\mathbf{0.695 \pm 0.053}$	0.777 ± 0.026	$\mathbf{0.793 \pm 0.029}$
+Both Suppression	$\mathbf{0.869 \pm 0.001}$	0.904 ± 0.062	0.675 ± 0.045	0.779 ± 0.022	0.787 ± 0.021

Table 1. The graph classification accuracy comparison on five datasets ("-" means not available, best results are shown in bold).

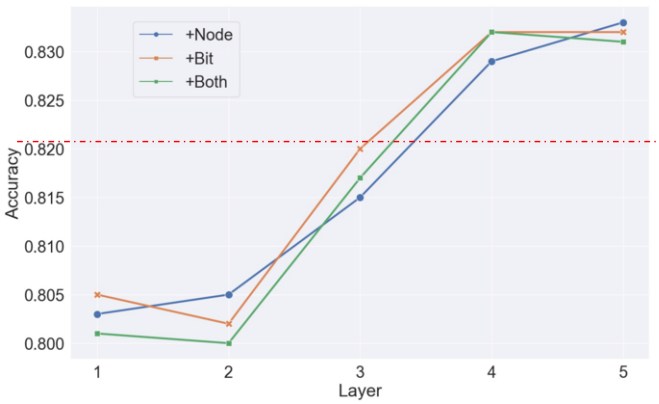

Figure 4. The ablation study of constraints on layer depth. Red dash line denotes the original accuracy (0.821).

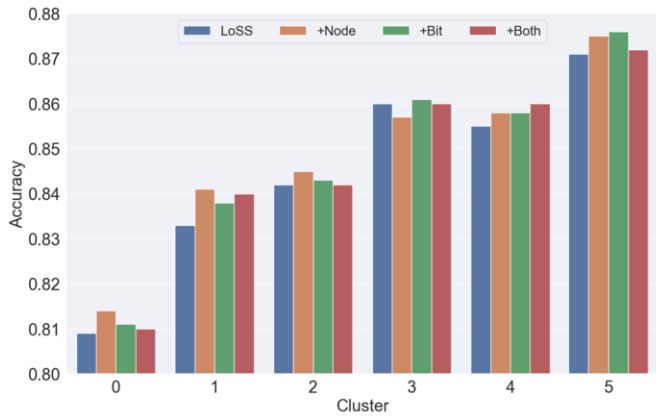

Figure 5. The ablation study of different cluster number impact for model performance.

influence of low-correlation suppression. The +Node-Level Suppression, +Bit-Level Suppression, and +Both Suppression are the results of our proposed model under the influence of node-level constraint, bit-level constraint, and both constraints, respectively.

As shown in Table 1, LoSS consistently outperforms all baselines on all datasets. In particular, LoSS achieves average accuracies of 79.3% and 69.5%, which are 2.0% and 3.4% improvement over the second-best ranked method. As a matter of fact, the results of our proposed model without the influence of low-correlation suppression have already outperformed the other comparing methods. It shows that the probabilistic random walk clustering based decoupling and aggregation mechanism can effectively find useful features and reduce the interference of redundant features. Meanwhile, from the result of +Node, +Bit, and +Both, we can see that the performance of our model generally improved under the influence of low-correlation suppression. It verifies the effectiveness of the low-correlation suppression mechanism. Overall, the proposed LoSS shows promising results against comparing methods.

4.5 Qualitative Results

In this section, we provide the qualitative evaluations of three low-correlation suppression methods to show the influence of three low-correlation suppression methods on to model's decision mechanism.

Figure 6 shows the visual results of the synthetic dataset. We use the correlation analysis method, described in Section 3.3, to analyze the decision correlation of each node, and visualize some samples, which are corrected by low-correlation suppression. The synthetic dataset is generated by mixing predefined structures, and the label of the data is directly related to whether it has corresponding structures. Therefore, we can find out the label related part, which is tagged target in Figure 6. We can see that the focus of our model are transferred from the irrelevant nodes to the relevant nodes (i.e., target nodes), under the influence of low-correlation suppression.

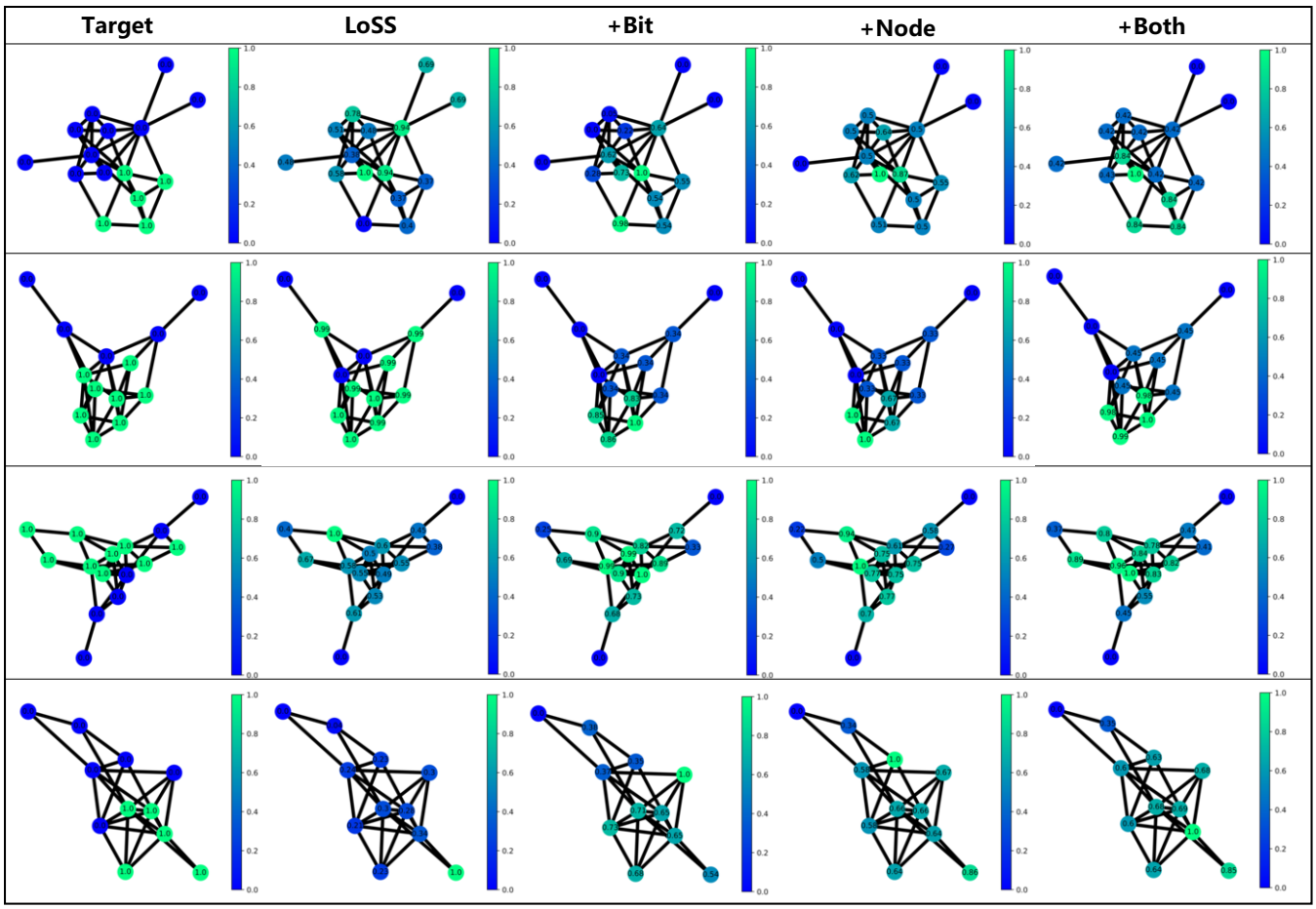

Figure 6. The visual results of synthetic dataset.

4.6 Ablation Study

To further analyze the model characteristics, this section first conducts the ablation study of different cluster numbers, and then the constraint layer depth impacts are evaluated.

The influence of constraint layer depth for different constraints is shown in Figure 4, where we can see that with the decrease of constraint depth, the gains of the three constraints are all gradually decreasing. The deep constraint layer brings model performance gains. On the other hand, the shallow constraint layer has a negative impact on the performance of the model. The possible reason is that the deep constraint layer could affect the final decision of the model more directly, and too many uncoupled useful features are lost in the shallow constraint layer.

Furthermore, we conduct the ablation study of different cluster number impacts for our model, which is shown in Figure 5. The traditional pair-wise graph can be regarded as a kind of special hypergraph, in which a hyperedge only contains two nodes. Thus, we use the original graph as the only input and get the result of cluster 0. On this basis, we introduce the probabilistic random walk clustering and gradually increase the number of clusters to decouple graph features. We can see that the model performance is greatly improved when the cluster is introduced. Moreover, the performance gain of the model gradually stabilizes as the number of clusters increases. And a more significant number of clusters will result in better model performance, which also requires more graph neural network units to handle each cluster.

5 Conclusion

In this paper, we proposed LoSS, a novel graph classification framework. LoSS overcomes the drawback of existing graph classification models that directly aggregate messages from the first-order neighbors of each node and mix useful and redundant features. Our method improves the performance of graph classification and is promising to be used in drug discovery research and social network analysis.

Acknowledgements

This work is funded by National Key Research and Development Project (Grant No: 2022YFB2703100), the Fundamental Research Funds for the Central Universities (2021FZZX001-23), Ningbo Natural Science Foundation (2022J182), the Starry Night Science Fund of Zhejiang University Shanghai Institute for Advanced Study (Grant No: SN-ZJU-SIAS-001), and the advanced computing resources provided by the Supercomputing Center of Hangzhou City University.

References

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek, 'On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation', *PloS one*, **10**(7), e0130140, (2015).

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun, 'Spectral networks and locally connected networks on graphs', *arXiv preprint arXiv:1312.6203*, (2013).

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst, 'Convolutional neural networks on graphs with fast localized spectral filtering', *Advances in Neural Information Processing Systems*, **29**, 3844–3852, (2016).

[4] 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, pp. 3558–3565, (2019).

[5] Zunlei Feng, Jiacong Hu, Sai Wu, Xiaotian Yu, Jie Song, and Mingli Song, 'Model doctor: A simple gradient aggregation strategy for diagnosing and treating cnn classifiers', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 616–624, (2022).

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec, 'Inductive representation learning on large graphs', in *Advances in Neural Information Processing Systems*, pp. 1024–1034, (2017).

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

[8] Yongcheng Jing, Yiding Yang, Xinchao Wang, Mingli Song, and Dacheng Tao, 'Amalgamating knowledge from heterogeneous graph neural networks', in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15709–15718, (2021).

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, 'Imagenet classification with deep convolutional neural networks', *Communications of the ACM*, **60**(6), 84–90, (2017).

[10] Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang, 'Dataset distillation via factorization', *Advances in Neural Information Processing Systems*, **35**, 1100–1113, (2022).

[11] Songhua Liu, Jingwen Ye, Runpeng Yu, and Xinchao Wang, 'Slimmable dataset condensation', in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3759–3768, (2023).

[12] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu, 'Disentangled graph convolutional networks', in *International conference on machine learning*, pp. 4212–4221. PMLR, (2019).

[13] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon, 'Higher-order explanations of graph neural networks via relevant walks', *IEEE transactions on pattern analysis and machine intelligence*, **44**(11), 7581–7596, (2021).

[14] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt, 'Weisfeiler-lehman graph kernels.', *Journal of Machine Learning Research*, **12**(9), (2011).

[15] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt, 'Efficient graphlet kernels for large graph comparison', in *Artificial intelligence and statistics*, pp. 488–495. PMLR, (2009).

[16] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje, 'Learning important features through propagating activation differences', in *International conference on machine learning*, pp. 3145–3153. PMLR, (2017).

[17] Jie Song, Yixin Chen, Xinchao Wang, Chengchao Shen, and Mingli Song, 'Deep model transferability from attribution maps', *Advances in Neural Information Processing Systems*, **32**, (2019).

[18] Yongduo Sui, Xiang Wang, Jiancan Wu, Min Lin, Xiangnan He, and Tat-Seng Chua, 'Causal attention for interpretable and generalizable graph classification', in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1696–1705, (2022).

[19] Mukund Sundararajan, Ankur Taly, and Qiqi Yan, 'Axiomatic attribution for deep networks', in *International conference on machine learning*, pp. 3319–3328. PMLR, (2017).

[20] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, 'Graph attention networks', *stat*, **1050**, 20, (2017).

[21] Cédric Vincent-Cuaz, Rémi Flamary, Marco Corneli, Titouan Vayer, and Nicolas Courty, 'Template based graph neural network with optimal transport distances', *Advances in Neural Information Processing Systems*, **35**, 11800–11814, (2022).

[22] Max Welling and Thomas N Kipf, 'Semi-supervised classification with graph convolutional networks', in *J. International Conference on Learning Representations (ICLR 2017)*, (2016).

[23] Junran Wu, Shangzhe Li, Jianhao Li, Yicheng Pan, and Ke Xu, 'A simple yet effective method for graph classification', *arXiv preprint arXiv:2206.02404*, (2022).

[24] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka, 'How powerful are graph neural networks?', *arXiv preprint arXiv:1810.00826*, (2018).

[25] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar, 'Hypergcn: A new method for training graph convolutional networks on hypergraphs', *Advances in neural information processing systems*, **32**, (2019).

[26] Pinar Yanardag and SVN Vishwanathan, 'Deep graph kernels', in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, (2015).

[27] Xingyi Yang, Jingwen Ye, and Xinchao Wang, 'Factorizing knowledge in neural networks', in *European Conference on Computer Vision*, pp. 73–91. Springer, (2022).

[28] Xingyi Yang, Daquan Zhou, Songhua Liu, Jingwen Ye, and Xinchao Wang, 'Deep model reassembly', *Advances in neural information processing systems*, **35**, 25739–25753, (2022).

[29] Yiding Yang, Zunlei Feng, Mingli Song, and Xinchao Wang, 'Factorizable graph convolutional networks', *arXiv preprint arXiv:2010.05421*, (2021).

[30] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji, 'Explainability in graph neural networks: A taxonomic survey', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2022).

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