Self-supervised Graph Neural Networks via Diverse and Interactive Message Passing

Liang Yang[1,2,*], Cheng Chen[1,*], Weixun Li[1,*], Bingxin Niu[1], Junhua Gu[1],
Chuan Wang[2,†], Dongxiao He[3], Yuanfang Guo[4], Xiaochun Cao[5]

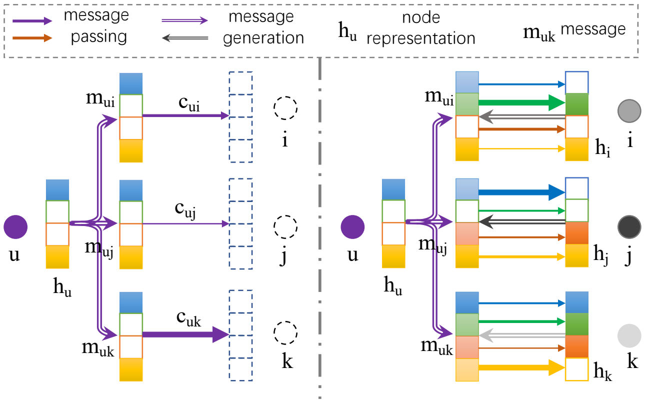
[1]School of Artificial Intelligence, Hebei University of Technology, Tianjin, China
[2]State Key Laboratory of Information Security, IIE, CAS, Beijing, China
[3]College of Intelligence and Computing, Tianjin University, Tianjin, China
[4]State Key Laboratory of Software Development Environment, Beihang University, China
[5]School of Cyber Science and Technology, Shenzhen Campus, Sun Yat-sen University, Shenzhen 518107, China
yangliang@vip.qq.com, {1437274010, 1043246925}@qq.com, wangchuan@iie.ac.cn

Abstract

By interpreting Graph Neural Networks (GNNs) as the message passing from the spatial perspective, their success is attributed to Laplacian smoothing. However, it also leads to serious over-smoothing issue by stacking many layers. Recently, many efforts have been paid to overcome this issue in semi-supervised learning. Unfortunately, it is more serious in unsupervised node representation learning task due to the lack of supervision information. Thus, most of the unsupervised or self-supervised GNNs often employ *one-layer GCN* as the encoder. Essentially, the over-smoothing issue is caused by the over-simplification of the existing message passing, which possesses two intrinsic limits: blind message and uniform passing. In this paper, a novel Diverse and Interactive Message Passing (DIMP) is proposed for self-supervised learning by overcoming these limits. Firstly, to prevent the message from blindness and make it interactive between two connected nodes, the message is determined by both the two connected nodes instead of the attributes of one node. Secondly, to prevent the passing from uniformness and make it diverse over different attribute channels, different propagation weights are assigned to different elements in the message. To this end, a natural implementation of the message in DIMP is the element-wise product of the representations of two connected nodes. From the perspective of numerical optimization, the proposed DIMP is equivalent to performing an overlapping community detection via expectation-maximization (EM). Both the objective function of the community detection and the convergence of EM algorithm guarantee that DMIP can prevent from over-smoothing issue. Extensive evaluations on node-level and graph-level tasks demonstrate the superiority of DIMP on improving performance and overcoming over-smoothing issue.

Introduction

Originating from spectral graph theory (Chung and Graham 1997), Graph Neural Networks (GNNs) show superior performance on modeling ubiquitous irregular data in many fields, such as computer vision, natural language processing and information retrieval, etc (Wu et al. 2021). By reducing the computational complexity of graph Fourier transformation via approximation, most GNNs, especially Graph Convolutional Networks (GCN) (Kipf and Welling 2017) and its variants including SGC (Wu et al. 2019) actually perform Laplacian smoothing (Li, Han, and Wu 2018) and low-passing filtering (Wu et al. 2019). By interpreting them as the message passing from the spatial perspective, many methods have been proposed (Zhou et al. 2020). Although methods based on the intuitive message passing achieve



(a) Uniform and Blind Message Passing

(b) Diverse and Interactive Message Passing

Figure 1: Comparison between exiting uniform and blind message passing and the proposed diverse and interactive message passing (DIMP). $\mathbf{h}_u$'s denote the node representations, while $\mathbf{m}_{ui}$'s stand for the messages. Arrows with single line denote the passing of the message, and the thickness stands for the propagation weight. The arrows with double lines stand for the generation of message. The existing uniform and blind message passing generates messages from the representation of one node, i.e., u, and propagates them with uniform weights. In contrary, our proposed diverse and interactive message passing generates messages from two connected nodes and propagates them with diverse weights.

[*]These authors contributed equally.

[†]Corresponding author.

state-of-the-art on semi-supervised node classificaiton (Zhu et al. 2021b; Tao et al. 2020), it leads to serious over-smoothing issue by stacking many layers (Li, Han, and Wu 2018; Oono and Suzuki 2020; Xu et al. 2018). To overcome this issue in semi-supervised learning, many efforts have been paid (Klicpera, Bojchevski, and Günnemann 2019; Xu et al. 2018; Zhao and Akoglu 2020; Rong et al. 2020; Feng et al. 2020). For example, DropEdge randomly drops some edges in each training epoch and constrains nodes be correctly classified (Rong et al. 2020), while GRAND randomly drops some nodes to form some different views and constrains corresponding node in different views possesses similar representations, which can be used to correctly classify nodes.

Unfortunately, the over-smoothing issue is more serious in unsupervised node representation learning task compared to semi-supervised node classification task due to the lack of supervision information. Thus, most of the unsupervised or self-supervised GNNs often employ *one-layer GCN* as the encoder, and only focus on the design of the decoder or self-supervised objective function (Kipf and Welling 2016; Zhu et al. 2020b; You et al. 2020; Peng et al. 2020). The direct drawback of one-layer GCN is the incapability in capturing multi-scale topology information. To alleviate this drawback, many attempts have been made by designing different contrastive learning algorithms. DGI (Velickovic et al. 2019) incorporates global information by contrasting between node and graph representations, SubG-Con (Jiao et al. 2020) combines mesoscopic information by contrasting between node and subgraph representations, and InfoMotif (Sankar et al. 2020) preserves high-order information by using motif structure. However, it is ineffective to incorporate multi-scale topology into the one-layer GCN encoder via the self-supervised objective function, since the only trainable component in the one-layer GCN is the mapping function, which is implemented by a fully-connected layer.

Therefore, this paper aims at a specific graph neural network for self-supervised node representation learning. It can both incorporate multi-scale information and effectively prevent over-smoothing issue without supervision information. To this end, the existing message passing framework is investigated (in Section). Actually, it possesses two intrinsic limits as shown in Fig. (1)(a). (1) *The message is blind.* That is, the messages propagated to all the neighbourhoods are the same by ignoring the characteristic of different neighbourhoods. (2) *The passing is uniform.* That is, all the elements in one message are propagated via the same weights by ignoring the characteristic of different attributes. These two simplifications ignore the complicated relationship between nodes, and thus cause the representations of all nodes be too similar to be distinguished from each other after many message passing steps, i.e., over-smoothing issue.

To overcome these two limits in existing message passing, a novel Diverse and Interactive Message Passing (DIMP) is proposed for self-supervised learning as shown in Fig. (1)(b) (in Section). Firstly, to prevent the message from blindness and make it interactive between two connected nodes, the message is determined by both two connected nodes instead of the attributes of one node. Secondly, to prevent the passing from uniformness and make it diverse over different attribute channels, different propagation weights should be assigned to different elements in the message. To this end, a natural implementation of the message in DIMP is the element-wise product of the representations of two connected nodes. It not only makes different neighbourhoods obtain different messages, but also indicates different weights for different channels. From the perspective of numerical optimization, the proposed DIMP is equivalent to performing an overlapping community detection (Ball, Karrer, and Newman 2011) via expectation-maximization (EM) (Moon 1996). Both the objective function of the community detection and the convergence of EM algorithm guarantee that DMIP can prevent from over-smoothing issue. The main contributions of this paper are summarized as follows.

- The drawbacks of existing message passing, i.e., blind message and uniform passing, are investigated.
- A novel Diverse and Interactive Message Passing (DIMP) is proposed for self-supervised learning tasks.
- The ability of DIMP on preventing over-smoothing issue is theoretically analyzed and experimentally evaluated.
- DIMP achieves new state-of-the-art on node-level and graph-level self-supervised learning tasks.

Notations and Preliminaries

Notations Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph with node set $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ and edge set $\mathcal{E}$, where N is the number of nodes. The topology of graph $\mathcal{G}$ can be represented by its adjacency matrix $\mathbf{A} = [a_{ij}] \in \{0,1\}^{N\times N}$, where $a_{ij} = 1$ if and only if there exists an edge $e_{ij} = (v_i, v_j)$ between nodes v_i and v_j. The degree matrix $\mathbf{D}$ is a diagonal matrix with diagonal element $d_i = \sum_{i=1}^{N} a_{ij}$ as the degree of node v_i. $\mathcal{N}(v_i) = \{v_j | (v_i, v_j) \in \mathcal{E}\}$ stands for the neighbourhoods of node v_i. $\mathbf{X} \in \mathbf{R}^{N\times F}$ and $\mathbf{H} \in \mathbf{R}^{N\times F'}$ denote the collections of node attributes and representations with the i^{th} rows, i.e., $\mathbf{x}_i \in \mathbb{R}^F$ and $\mathbf{h}_i \in \mathbb{R}^{F'}$, corresponding to node v_i, where F and F' stand for the dimensions of attribute and representation.

Message Passing Framework Most Graph Neural Networks (GNNs) follow an aggregation-combination strategy (Gilmer et al. 2017), where each node representation is iteratively updated by aggregating node representations from neighbourhoods and combining the aggregated representation with the node representation itself as follows

$$\bar{\mathbf{h}}_v^k = \mathrm{AGGREGATE}^k\left(\left\{\mathbf{h}_u^{k-1} | u \in \mathcal{N}(v)\right\}\right), \quad (1)$$

$$\mathbf{h}_v^k = \mathrm{COMBINATE}^k\left(\mathbf{h}_v^{k-1}, \bar{\mathbf{h}}_v^k\right), \quad (2)$$

where $\bar{\mathbf{h}}_v^k$ stands for the aggregated representation from neighbourhoods. Except for the concatenation, such as GraphSAGE (Hamilton, Ying, and Leskovec 2017) and H2GCN (Zhu et al. 2020a), average (or summation) is widely adopted as the implementation of $\mathrm{COMBINATE}^k(\cdot,\cdot)$ function, such as GCN (Kipf and Welling 2017), GAT (Velickovic et al. 2018) and GIN (Xu et al. 2019) etc. Except for the MAX and LSTM implementations in GraphSAGE (Hamilton, Ying, and Leskovec

2017), most GNNs utilize average function as the implementation of $\mathrm{AGGREGATE}^k$. Therefore, most GNNs can be unified under the following formula

$$\mathbf{h}_v^k = \sigma\Big(\big(c_{vv}^k \mathbf{h}_v^{k-1} + \sum_{u \in \mathcal{N}(v)} c_{uv}^k \mathbf{h}_u^{k-1}\big)\mathbf{W}^k\Big), \tag{3}$$

where $\mathbf{W}^k$ is the learnable parameters and the $\sigma(\cdot)$ is the nonlinear mapping function. The scalar c_{uv} is the averaging weight. For example, GCN (Kipf and Welling 2017) sets $c_{uv}^k = 1/\sqrt{(d_u+1)(d_v+1)}$, GIN (Xu et al. 2019) sets $c_{uv}^k = 1$ for $u \neq v$ and $c_{vv}^k = 1 + \epsilon^k$, and GAT (Velickovic et al. 2018) learns non-negative c_{uv}^k using attention mechanism. Recently, to handle the network with heterophily via high-passing filtering, GPRGNN (Chien et al. 2021) sets $c_{uv}^k = \gamma^k/\sqrt{(d_u+1)(d_v+1)}$ with γ^k as learnable real number while FAGCN (Bo et al. 2021) directly relaxes the learnable c_{uv} in GAT to real number.

Limits of Existing Message Passing

Note that there are two serious and inherent limits of the existing message passing framework as shown in Fig. (1)(a).

- **The message is blind.** As shown in Eq. (1), the node representation itself, i.e., $\mathbf{h}_u^{k-1}$, is adopted as the messages to be propagated to all its neighbourhoods. This propagation scheme ignores the specific characteristic of different neighbourhoods. That is, message is blind to know the nodes, which it will be propagated to. Actually, different neighbourhods often possess different characteristics, and should obtain different messages. Thus, existing message passing can be regarded as a simplification with blind message.
- **The passing is uniform.** As shown in Eq. (3), all the elements in message $\mathbf{h}_u^{k-1}$ are propagated to one node v with the same weights, i.e., c_{uv}^k. That is, all elements in the message possess the uniform importance over one propagation. In fact, different node attributes, i.e., different elements in a message, may have different influence to neighbourhoods, and should be set with different propagation weights. Thus, existing message passing can be seen as a simplification with uniform propagation weight.

Therefore, existing message passing can be seen as a simplification with blind message and uniform propagation weight. This simplifications ignores the complicated relationship between nodes. Thus, they may cause the representations of all nodes be too similar to be distinguished from each other after many message passing steps, i.e., over-smoothing issue.

Methodology

This section elaborates the proposed Diverse and Interactive Message Passing (DIMP) and interprets it from the perspective of numerical optimization to reveal its ability on preventing over-smoothing issue.

Diverse and Interactive Message Passing

This section considers how to alleviate the limits of traditional message passing mentioned in Section . Thus, the diverse and interactive message passing is proposed as shown in Fig. (1)(b). To prevent the message from blindness, the message in the aggregation (Eq. (1) and the unified formula (Eq. (3)), i.e., $\mathbf{h}_u^{k-1}$, should be improved to take both the connected nodes into consideration. Thus, they can be reformulated as

$$\bar{\mathbf{h}}_v^k = \mathrm{AGGREGATE}^k\left(\{\bar{\mathbf{m}}_{uv}^k | u \in \mathcal{N}(v)\}\right), \tag{4}$$

$$\mathbf{h}_v^k = \sigma\Big(\big(c_{vv}^k \bar{\mathbf{m}}_{vv}^k + \sum_{u \in \mathcal{N}(v)} c_{uv}^k \bar{\mathbf{m}}_{uv}^k\big)\mathbf{W}^k\Big), \tag{5}$$

where $\bar{\mathbf{m}}_{uv}^k$ represents the message propagated from node u to node v, and should be determined by both nodes. $\bar{\mathbf{m}}_{vv}^k$ denotes the message from node v to itself along self-loop edge. Besides, to prevent the passing from uniformness, the passing in the Eq. (5) should be extended to arranging different weights to different channels, i.e.,

$$\mathbf{h}_v^k = \sigma\Big(\big(\mathbf{c}_{vv}^k \odot \bar{\mathbf{m}}_{vv}^k + \sum_{u \in \mathcal{N}(v)} \mathbf{c}_{uv}^k \odot \bar{\mathbf{m}}_{uv}^k\big)\mathbf{W}^k\Big), \tag{6}$$

where $\odot$ stands for the element-wise product. $\mathbf{c}_{uv}^k$ is the propagation weight vector which has the same length as $\bar{\mathbf{m}}_{uv}^k$. Each element in $\mathbf{c}_{uv}^k$ represents the propagation weight corresponding to the element in the message $\bar{\mathbf{m}}_{uv}^k$. For simplicity, the propagation weight $\mathbf{c}_{uv}^k$ can be integrated into the message $\bar{\mathbf{m}}_{uv}^k$ to form the unified message $\mathbf{m}_{uv}^k = \mathbf{c}_{uv}^k \odot \bar{\mathbf{m}}_{uv}^k$. Thus, Eq. (6) can be rewritten as

$$\begin{aligned} \mathbf{h}_v^k &= \sigma\Big(\big(\mathbf{m}_{vv}^k + \sum_{u \in \mathcal{N}(v)} \mathbf{m}_{uv}^k\big)\mathbf{W}^k\Big) \\ &= \sigma\Big(\big(\sum_u a_{uv}\mathbf{m}_{uv}^k\big)\mathbf{W}^k\Big). \end{aligned} \tag{7}$$

The remaining issue is the design of message $\mathbf{m}_{uv}^k$. The key idea is to make message $\mathbf{m}_{uv}^k$ reflect the characteristic of both nodes. Since only the representations of the two connected nodes, i.e., $\mathbf{h}_v^{k-1}$ and $\mathbf{h}_u^{k-1}$, are known, it is natural to form the message by combining these two representations. The candidate combination operations include summation, concatenation and element-wise product.

- **Summation.** Although the summation $\mathbf{m}_{uv}^k = \mathbf{h}_v^{k-1} + \mathbf{h}_u^{k-1}$ prevents the message from blindness, by adopting this operation, the Eq. (7) degrades to the traditional message passing with blind message in Eq. (3) .
- **Concatenation.** By adopting concatenation $\mathbf{m}_{uv}^k = [\mathbf{h}_v^{k-1} || \mathbf{h}_u^{k-1}]$, the length of the message may exponentially increase as the message passing. It makes the mapping matrix $\mathbf{W}^k$ become larger and larger, and thus tends to be over-fitting with limited labelled data.
- **Element-wise Product.** The element-wise product of the two representations can be formulated as

$$\mathbf{m}_{uv}^k = \frac{\mathbf{h}_v^{k-1} \odot \mathbf{h}_u^{k-1}}{< \mathbf{h}_v^{k-1}, \mathbf{h}_u^{k-1} >}, \tag{8}$$

where $\odot$ and $< \cdot, \cdot >$ represent the element-wise product and the inner-product, respectively. The inner-product of the representations in the denominator is to normalizes the message. The message obtained from element-wise product possesses many attractive characteristic. First, it has the same dimension as the node feature. Second, it will not degrade to message passing with blind message. Third, different propagation channels possess diverse propagation scheme, since each element in the message is determined by the corresponding two elements in the representations of two nodes. Note that, the element-wise product also indicates the potential to prevent over-smoothing due to the diverse propagation scheme.

Therefore, the proposed diverse and interactive message passing (DIMP) employs the element-wise product of the representations as the message. Thus, the DIMP iterates between Eqs. (7) and (8). Similar to GCN (Kipf and Welling 2017), the node attribute $\mathbf{x}_u$ is employed as the initial representations, i.e., $\mathbf{h}_u^0 = \mathbf{x}_u$, and each layer of DIMP consists of a step of message construction (Eq.(8) and a step of passing (Eq. (7). By stacking K layers of DIMP, i.e., k iteration between Eqs. (7) and (8), the final node representations can be obtained as $\mathbf{h}_u^K$'s.

Remark: DIMP in Eqs. (7) and (8) has the same number of parameters as the uniform and blind message passing in Eq. (3). Thus, their model complexities are the same.

Objective Function

This section provides the objective function employed to train the DIMP, i.e., the parameters in Eq. (7). Since the main task is to design a novel GNN for self-supervised learning, the infomax-based objective function utilized in DGI (Velickovic et al. 2019) is also adopted. It maximizes the mutual information between the patch representations and graph representation as

$$\mathcal{L} = \sum_{u=1}^{N} \log \mathcal{D}\left(\mathbf{h}_u^K, \mathbf{s}\right) + \sum_{v=1}^{M} \log\left(1 - \mathcal{D}\left(\tilde{\mathbf{h}}_v^K, \mathbf{s}\right)\right), \quad (9)$$

where $\mathbf{s} = \frac{1}{N}\sum_{u=1}^{N} \mathbf{h}_u^K$ denotes the representation of graph, and $\tilde{\mathbf{h}}_v^K$'s are the negative samples of the node representations, which are generated by performing DIMP on graph with randomly shuffled node attributes as in DGI (Velickovic et al. 2019). $\mathcal{D}\left(\mathbf{h}_u^K, \mathbf{s}\right)$ is the discriminator, and a simple bilinear scoring function is employed as

$$\mathcal{D}\left(\mathbf{h}_u^K, \mathbf{s}\right) = \sigma\left(\left(\mathbf{h}_u^K\right)^T \bar{\mathbf{W}}\mathbf{s}\right), \quad (10)$$

where $\bar{\mathbf{W}}$ is the learnable parameter for the discriminator and $\sigma(\cdot)$ stands for the logistic sigmoid nonlinearity. Maximizing the objective function in Eq. (9) is equivalent to maximizing the discriminator score between positive sample $\mathbf{h}_u^K$ and the graph representation $\mathbf{s}$ as well as minimizing the discriminator score between negative sample $\tilde{\mathbf{h}}_v^K$ and the graph representation $\mathbf{s}$.

Remark: Except for training DIMP in a self-supervised manner, DIMP in Eqs. (7) and (8) can also be trained in a semi-supervised manner by employing the cross-entropy loss as objective function as other semi-supervised GNNs. However, in order to highlight the advantages of diverse and interactive message passing instead of training the learnable parameters, this paper mainly focuses on the self-supervised training. Section shows that compared with some semi-supervised GNNs, self-supervised DIMP achieves comparable or even better performance.

Numerical Optimization Perspective

Recently, some attempts unify GNNs from the perspective of numerical optimization (Ma et al. 2020; Zhu et al. 2021a; Yang et al. 2021). They show that the message passing in GNNs, such as GCN (Kipf and Welling 2017), actually is the gradient descent of the graph Laplacian regularization with node attribute as the initial, i.e.,

$$tr(\mathbf{H}^T\tilde{\mathbf{L}}\mathbf{H}) = \frac{1}{2}\sum_{uv} a_{ij} \left\| \frac{\mathbf{h}_u}{\sqrt{d_u+1}} - \frac{\mathbf{h}_v}{\sqrt{d_v+1}} \right\|^2. \quad (11)$$

That is, each graph convolutional layer is equivalent to one gradient descent of Eq. (11). Since the solution to Eq. (11) is $\mathbf{h}_u = \sqrt{d_u+1}\mathbf{1}$, where $\mathbf{1}$ is the vector of ones, the obtained node representation from many graph convolutional layers is also $\mathbf{h}_u = \sqrt{d_u+1}\mathbf{1}$, which is only determined by the degree of node. Thus, GCN tends to be over-smoothing by stacking multiple layers. To prevent over smoothing issue, many GNNs, such as APPNP (Klicpera, Bojchevski, and Günnemann 2019), GCNII (Chen et al. 2020) and JKNet (Xu et al. 2018), balance the Eq. (11) with node attributes $\mathbf{X}$ as

$$\|\mathbf{H} - \mathbf{X}\|_F^2 + \lambda tr(\mathbf{H}^T\tilde{\mathbf{L}}\mathbf{H}), \quad (12)$$

where λ is the weighting parameter. Although the trick of balancing can alleviate the over-smoothing issue, it doesn't essentially solve the over-smoothing problem caused by the graph Laplacian regularization in Eq. (11).

Therefore, to essentially prevent the over-smoothing issue in massage passing, the key is to replace the graph Laplacian regularization with a novel one, whose solution can capture complicated topology information instead of the degree of node. Actually, the following theorem shows that the proposed DIMP in Section is equivalent to overlapping community detection (Ball, Karrer, and Newman 2011).

Theorem 1. *The diverse and interactive message passing in Eqs. (7) and (8) is equivalent to the expectation-maximization (EM) algorithm to maximize the likelihood of generating graph from community structure* $\{\mathbf{h}_u\}_{u=1}^N$ *via Poisson distribution in (Ball, Karrer, and Newman 2011) as follows*

$$\begin{aligned} P\left(\mathcal{G}\middle|\{\mathbf{h}_u\}_{u=1}^N\right) &= \prod_{u<v} \frac{\left(\mathbf{h}_u^T\mathbf{h}_v\right)^{a_{uv}}}{a_{uv}!}\exp\left(-\mathbf{h}_u^T\mathbf{h}_v\right) \quad (13) \\ &\times \prod_{u} \frac{\left(\frac{1}{2}\mathbf{h}_u^T\mathbf{h}_u\right)^{a_{uv}/2}}{(a_{uu}/2)!}\exp\left(-\frac{1}{2}\mathbf{h}_u^T\mathbf{h}_u\right). \end{aligned}$$

Proof. Maximizing the likelihood in Eq. (13) can be achieved by maximizing its log-likelihood as

$$\log P\left(\mathcal{G}\middle|\{\mathbf{h}_u\}_{u=1}^N\right) = \sum_{uv} a_{uv}\log\left(\mathbf{h}_u^T\mathbf{h}_v\right) - \sum_{uv}\mathbf{h}_u^T\mathbf{h}_v. \quad (14)$$

Direct maximization of this expression by differentiating are hard to solve. This difficulty can be overcome by employ EM algorithm. EM can be implemented via Jensen's inequality

$$\log\left(\sum_z x_z\right) \geq \sum_z q_z \log \frac{x_z}{q_z}, \tag{15}$$

where the exact equality can be achieved by making the specific choice $q_z = x_z / \sum_z x_z$. By apply Eq. (15) to the log-likelihood in Eq. (14), it gets

$$\log P\left(\mathcal{G} | \{\mathbf{h}_u\}\right) \geq \sum_{uvz} \left[a_{uv} q_{uv}(z) \log \frac{h_{uz} h_{vz}}{q_{uv}(z)} - h_{uz} h_{vz}\right], \tag{16}$$

and the exact equality can be achieved by setting

$$q_{uv}(z) = \frac{h_{uz} h_{vz}}{\sum_z h_{uz} h_{vz}} = \frac{h_{uz} h_{vz}}{< \mathbf{h}_u, \mathbf{h}_v >}. \tag{17}$$

Given the optimal values of the $q_{uv}(z)$, the optimal h_{uz} can be found by differentiating Eq. (16), which gives

$$h_{uz} = \frac{\sum_v a_{uv} q_{uv}(z)}{\sum_v h_{vz}}. \tag{18}$$

Thus, by iterating between Eqs. (17) and (18), Eq. (14) can be minimized, and the community structure can be obtained from $\mathbf{h}_u$'s.

It can be observed that the Eqs. (17) and (18) are the element-wise form of the message and passing in Eqs. (8) and (7), respectively. Thus, DIMP is equivalent to the EM algorithm to maximize Eq. (13). □

Note that compared to the smoothing effect of graph Laplacian regularization in Eq. (11), which only makes $\mathbf{h}_u$'s similar, maximizing Eq. (13) makes $\mathbf{h}_u$'s capture complicated overlapping community structure. In network science, the community is the mesoscopic structure of the graph compared to the local degree information, which determines the solution to minimizing graph Laplacian regularization. By maximizing Eq. (13), the obtained $\mathbf{h}_u$'s stand for the categorical probability distributions of nodes. $\mathbf{h}_u$'s can be employed to represent the overlapping community structure, since they are distributed representations instead of one-hot vectors. Specifically, the z^{th} element in $\mathbf{h}_u$, i.e., h_{uz}, represents the probability of node u belonging to community z. The probability distribution $\mathbf{h}_u$ actually captures the role of node in the network. The categorical distribution $\mathbf{h}_u$ of node u tends to concentrate on one community if it is the hub, and the categorical distribution $\mathbf{h}_u$ of node u distributed to many communities if it is at the boundary of community. Therefore, the obtained $\mathbf{h}_u$'s from Eq. (13) and DIMP are diverse and may not concerntrate to few points. Thus, the DIMP, which is equivalent to the EM algorithm to Eq. (13, can prevent from over-smoothing issue.

Evaluations

Datasets: Statistics of datasets used for node-level tasks are shown in Table 1. These datasets can be divided into three categories. **Citation Networks.** Cora, Citeseer, and Pubmed, which are widely used to verify GNNs, are standard citation network benchmark datasets (Sen et al. 2008; Namata et al. 2012). **Co-purchase Networks.** Amazon-C and Amazon-P are two networks of co-purchase relationships (Shchur et al. 2019). **Coauthor Networks.** Coauthor-CS and Coauthor-P are co-author networks based on the Microsoft Academic Graph from the KDD Cup 2016 challenge. Statistics of datasets used for graph-level tasks are shown in Table 2. Note that, the statistics of the numbers of nodes and edges are the average results over all graphs, since each datasets contains multiple graphs with different sizes. These datasets are from (Yanardag and Vishwanathan 2015).

Table 1: Statistics of datasets used for node-level tasks.

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
Amazon-C	13,752	245,861	767	10
Amazon-P	7,650	119,081	745	8
Coauthor-CS	18,333	81,894	6,805	15
Coauthor-P	34,493	247,962	8,415	5

Table 2: Statistics of datasets used for graph-level task.

Dataset	#Graphs	#Nodes	#Edges	#Classes
MUTAG	188	17.93	19.79	2
PTC-MR	344	14.29	14.69	2
IMDB-BIN	1,000	19.77	193.06	2
IMDB-MULTI	1,500	13.00	65.93	3
REDDIT-BIN	2,000	508.52	497.75	2

Baselines: For node-level tasks, most unsupervised and self-supervised GNNs are employed as the baselines. Unsupervised GNNs includes GAE and VGAE (Kipf and Welling 2016), ARGA and ARVAG (Pan et al. 2018), MGAE (Wang et al. 2017), GALA (Park et al. 2019), while self-supervised GNNs includes DGI (Velickovic et al. 2019), GMI (Peng et al. 2020), MVGRL (Hassani and Ahmadi 2020), GRACE (Zhu et al. 2020b), GCA (Zhu et al. 2021c), GraphCL (You et al. 2020) and SubG-Con (Jiao et al. 2020). To demonstrate the superiority of DIMP, except for the MLP, LogReg and LP, some state-of-the-art semi-supervised GNNs are also adopted, include Chebyshev (Defferrard, Bresson, and Vandergheynst 2016), GCN (Kipf and Welling 2017), GAT (Velickovic et al. 2018), SGC (Wu et al. 2019), MoNet (Monti et al. 2017). For clustering, except for the k-means and spectral clustering, DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), BigClam (Yang and Leskovec 2013) are used.

For graph-level task, i.e. graph classification, the baselines include DeepWalk and node2vec (**?**), sub2vec (Adhikari et al. 2018) and graph2vec (Narayanan et al. 2017), infograph (Sun et al. 2020), MVGRL and GraphCL. Similar to node classification task, semi-supervised GNNs are also employed, include GraphSAGE (Hamilton, Ying, and Leskovec 2017), GCN, GAT and GIN (Xu et al. 2019).

Table 3: Node Classification Results in Terms of Accuracy.

Method	Cora	CiteSeer	PubMed	Amazon-C	Amazon-P	Coauthor-CS	Coauthor-P
MLP	58.2±2.1	59.1±2.3	70.0±2.1	44.9±5.8	69.6±3.8	88.3±0.7	88.9±1.1
LogReg	57.1±2.3	61.0±2.2	64.1±3.1	64.1±5.7	73.0±6.5	86.4±0.9	86.7±1.5
LP	68.0±0.2	45.3±0.2	63.0±0.5	70.8±0.0	67.8±0.0	74.3±0.0	90.2±0.5
Chebyshev	81.2±0.5	69.8±0.5	74.4±0.3	62.6±0.0	74.3±0.0	91.5±0.0	92.1±0.3
GCN	81.5±0.2	70.3±0.3	**79.0±0.4**	76.3±0.5	87.3±1.0	**91.8±0.1**	**92.6±0.7**
GAT	**83.0±0.7**	**72.5±0.7**	**79.0±0.3**	79.3±1.1	86.2±1.5	90.5±0.7	91.3±0.6
MoNet	81.3±1.3	71.2±2.0	78.6±2.3	**83.5±2.2**	**91.2±1.3**	90.8±0.6	92.5±0.9
DGI	81.7±0.6	71.5±0.7	77.3±0.6	75.9±0.6	83.1±0.5	90.0±0.3	91.3±0.4
GMI	80.9±0.7	71.1±0.3	77.9±0.2	76.8±0.1	85.1±0.1	91.0±0.0	OOM
MVGRL	82.9±0.7	72.6±0.7	79.4±0.3	79.0±0.6	87.3±0.3	88.4±0.3	92.6±0.4
GRACE	80.0±0.4	71.7±0.6	79.5±1.1	71.8±0.4	81.8±1.0	90.1±0.8	92.3±0.6
GCA	81.0±0.4	71.9±0.5	80.5±1.1	80.8±0.4	87.1±1.0	91.3±0.4	93.1±0.3
SubG-Con	82.5±0.3	70.8±0.3	73.1±0.5	OOM	OOM	OOM	OOM
DIMP	**83.3±0.5**	**73.3±0.5**	**81.4±0.5**	**83.3±0.4**	**88.7±0.2**	**92.1±0.5**	**94.2±0.4**

Table 4: Node Clustering Results in Terms of NMI and ARI.

Methods	Cora		Citeseer		Pubmed	
	NMI	ARI	NMI	ARI	NMI	ARI
K-means	0.321	0.230	0.305	0.279	0.001	0.002
Spectral	0.127	0.031	0.056	0.010	0.042	0.002
BigClam	0.007	0.001	0.036	0.007	0.006	0.003
GraphEnc	0.109	0.006	0.033	0.010	0.209	0.184
DeepWalk	0.327	0.243	0.088	0.092	0.279	0.299
GAE	0.429	0.347	0.176	0.124	0.277	0.279
VGAE	0.436	0.346	0.156	0.093	0.229	0.213
MGAE	0.511	0.445	0.412	0.414	0.282	0.248
ARGA	0.449	0.352	0.350	0.341	0.276	0.291
ARVGA	0.450	0.374	0.261	0.245	0.117	0.078
GALA	0.577	0.511	0.441	0.446	0.327	0.321
MVGRL	0.572	0.495	0.469	0.449	0.322	0.296
DIMP	**0.581**	**0.522**	**0.471**	**0.471**	**0.346**	**0.328**

Implementation Details The proposed DIMP employs 4-layers message passing network. The parameters in the mapping function and the discriminator function are initialized using Xavier initialization and trained using Adam optimizer with an initial learning rate of 0.001. The number of epochs and batch size are chose from $[10, 20, 40, 100]$ and $[32, 64, 128, 256]$, respectively. Besides, early stopping with a patience of 20 is also utilized. For classification tasks, the parameter C of the SVM is chosen from $[10^{-3}, 10^{-2}, \cdots, 10^{2}, 10^{3}]$.

Node-level Tasks

Node Classification Task On citation networks, we use 20 labeled nodes per class as the training set, 20 nodes per class as the validation set, and the rest as the testing set as in (Yang, Cohen, and Salakhutdinov 2016). On co-purchase and co-author networks, we use 30 labeled nodes per class as the training set, 30 nodes per class as the validation set, and the rest as the testing set. For fair comparison, the performance of all the methods are obtained on the same splits.

Table 5: Graph Classification Results in Terms of Accuracy.

METHOD	MUTAG	PTC-MR	IMDB-B	IMDB-M	RDT-B
GraphSage	85.1±7.6	63.9±7.7	72.3±5.3	50.9±2.2	-
GCN	85.6±5.8	64.2±4.3	74.0±3.4	51.9±3.8	50.0±0.0
GIN	**89.4±5.6**	64.6±7.0	**75.1±5.1**	**52.3±2.8**	**92.4±2.5**
GAT	89.4±6.1	**66.7±5.1**	70.5±2.3	47.8±3.1	85.2±3.3
DeepWalk	83.7±1.5	57.9±1.3	50.7±0.3	34.7±0.2	-
node2vec	72.6±10.	58.6±8.0	-	-	-
sub2vec	61.1±15.	60.0±6.4	55.3±1.5	36.7±0.8	71.5±0.4
graph2vec	83.2±9.6	60.2±6.9	71.1±0.5	50.4±0.9	75.8±1.0
Infograph	89.0±1.1	61.7±1.4	73.0±0.9	49.7±0.5	82.5±1.4
MVGRL	89.7±1.1	62.5±1.7	74.2±0.7	51.2±0.5	84.5±0.6
GraphCL	86.8±1.3	-	71.1±0.4	-	89.5±0.8
DIMP	**91.5±1.1**	**64.2±1.2**	**74.8±0.8**	**52.0±0.6**	**91.9±0.6**

Following DGI (Velickovic et al. 2019) the mean classification accuracy with standard deviation on the test nodes after 50 runs of training is reported in Table 3. The results with bold font represent the best performance. For better illustration, the best performance of semi-supervised and self-supervised GNNs are marked, respectively. It can be observed that our proposed DIMP significantly and consistently outperforms other self-supervised baselines, and achieves comparable results as the best semi-supervised GNNs. Note that the advantages of DIMP is more remarkable on large networks, such as Pubmed, Amazon-Computers and Coauthor-Physics. On some large network, such as Pubmed, Coauthor-CS and Coauthor-Physics, the proposed self-supervised DIMP outperform best semi-supervised GNNs. These notable performance improvements demonstrate the superiority of diverse and interactive message passing on capturing complicated relationship between connected nodes compared to the existing uniform and blind message passing.

Node Clustering Task Node clustering task is conduced by employing k-means on the obtained node representations. The clustering results averaged over 50 runs in terms

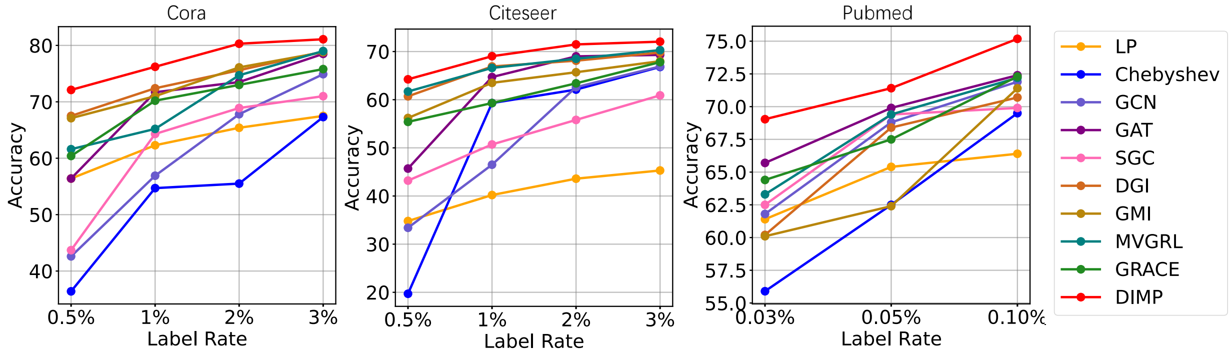


Figure 2: Node classification results with limited training labels on Cora, Citeseer and Pubmed.

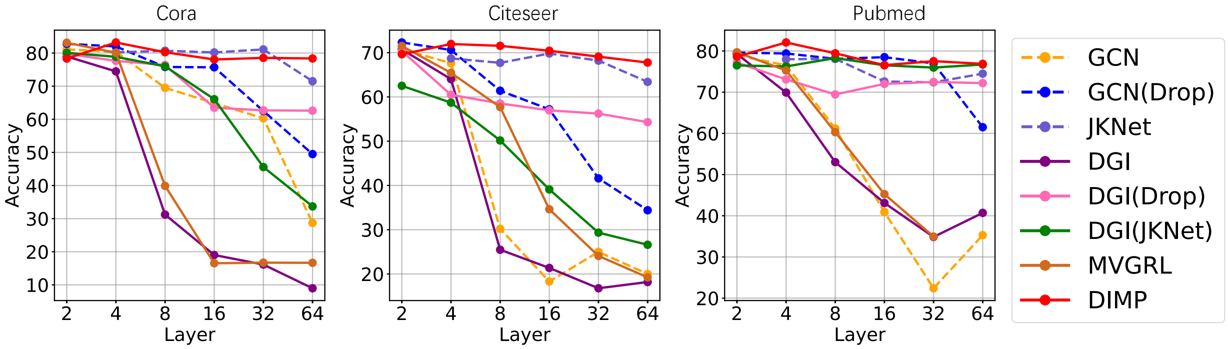


Figure 3: Node classification results of GNNs with various depth on Cora, Citeseer and Pubmed. Dashed lines denote semi-supervised GNNs, while solid lines stand for self-supervised ones.

of NMI and ARI are shown in Table 4. It shows that DIMP outperforms other baselines. Besides, the improvements are more significant in terms of ARI compared to those in terms of NMI. This indicates that DIMP tends to capture complete and comprehensive information by enhancing the scheme of message passing.

Limited Labeled Training Data To show the impact of label rate on performance, the performance of node classification with limited labeled training data are reported in Fig. 2. It shows that the improvements of DIMP are consistent with different label rates, and the improvements are more remarkable with lower label rate, e.g. 0.5%, on Cora and Citerseer. This demonstrates that the DIMP can effectively explore the high-order interaction between nodes, which plays the role of self-supervision in additional to the given supervision information.

Above three node-level experiments demonstrate the superiority of DIMP on node representation learning by exploring and leveraging complicated relationship between nodes from local and global context.

Graph-level Tasks

Following InfoGraph (Sun et al. 2020) the graph embedding can be obtained by averaging all embedding of nodes in the graph. The mean 10-fold cross validation accuracy with standard deviation after 5 runs followed by a linear SVM is reported in Table 5. It shows that DIMP achieves better graph classification accuracy compared to other unsupervised and self-supervised methods, especially the SOTA based on contrastive learning, infogrpah (Sun et al. 2020), MVGRL (Hassani and Ahmadi 2020) and GraphCL (You et al. 2020), and achieves the comparable performance as the semi-supervised GNNs. Since the graph representation is the average of node representations, a good graph representation should be induced by the semantic distribution of nodes in the embedding space. Actually, DIMP rearranges the nodes by effectively exploring the complicated relationship between them via flexible message passing. Therefore, the better graph representation verifies the effectiveness and the flexibility of the proposed DIMP.

Preventing Over-smoothing Issue Different from existing self-supervised GNNs, which adopted one-layer GCN as encoder, the proposed DIMP employs 4 diverse and interactive message passing layer. This contributes some performance improvements, since it possesses the potential to explore and capture global information. In this section, its ability on preventing over-smoothing is investigated. Fig. 3 shows the node classification accuracies with various depth on Cora and Citeseer. The depths of GNNs range in 2, 4, 8, 16, 32 and 64. For better illustration, self-supervised methods and semi-supervised methods are represented in solid lines and dashed lines, respectively.

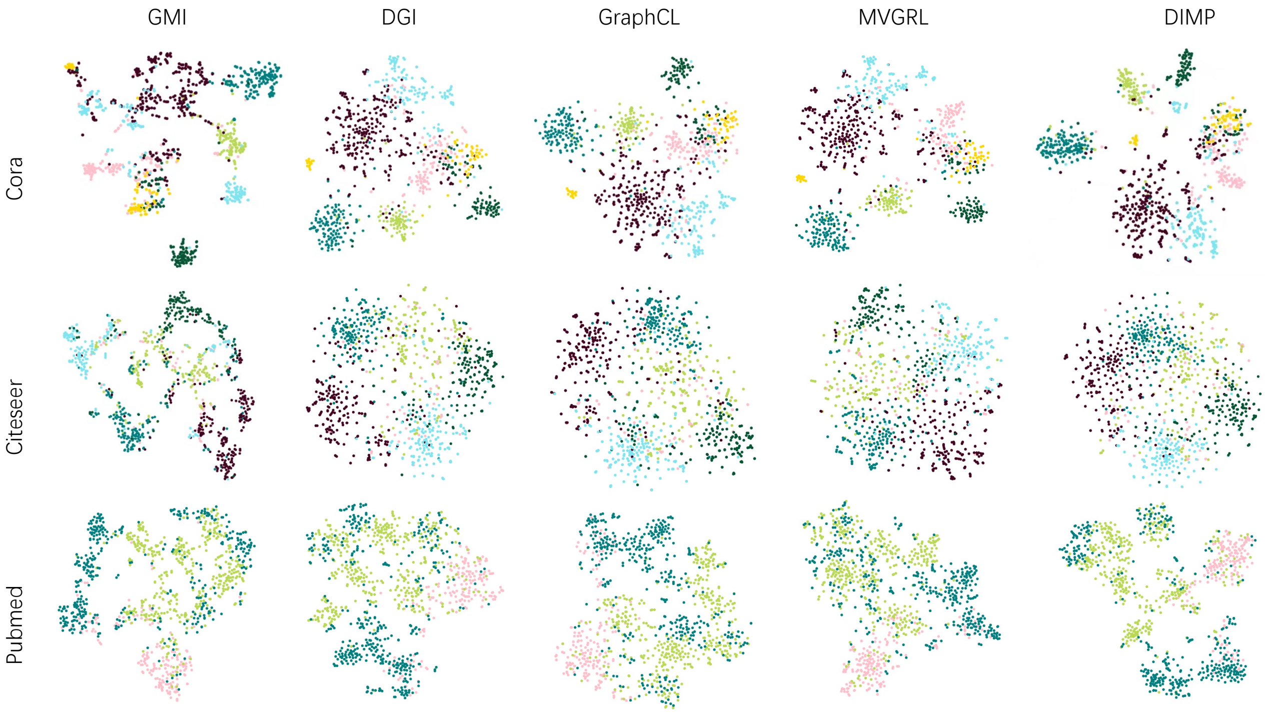


Figure 4: t-SNE of the node representations obtained from different GNNs (GMI (Peng et al. 2020), DGI (Velickovic et al. 2019), GraphCL (You et al. 2020), MVGRL (Hassani and Ahmadi 2020) and the proposed DIMP) on the Cora, Citeseer and Pubmed datasets.

It can be observed that GCN (semi-supervised) and DGI (self-supervised) tend to be over-smoothing as depth increase. Although DropEdge and JKNet can effectively alleviate over-smoothing issue for semi-supervised GNNs, their effects are weakened for self-supervised ones, especially the JKNet. The proposed DIMP performs better than others on overcoming over-smoothing issue in self-supervised learning. Thus, the proposed DIMP possesses the ability on exploring global information without over-smoothing issue, and thus is more powerful for self-supervised learning.

Visualizations To provide an intuitive understanding of DIMP, the t-SNE of the node embeddings obtained from different GNNs, include GraphCL, MVGRL and the proposed DIMP on the Cora, Citeseer and Pubmed is shown in Fig. 4. The colors of nodes represent their labels. It is obvious that the distribution of node representations obtained from DIMP meets that of the node label much better than others. The discriminability reflects the characteristic that nodes in the same class concentrate while nodes in different classes disperse. The proposed DIMP achieves promotions on both of Cora and Citeseer. Note that distribution of representation form self-supervised DIMP on Cora possesses some attractive visual characteristic as semi-supervised GNNs.

Conclusions

To overcome the over-smoothing issue in unsupervised and self-supervised GNNs, two serious drawbacks of existing message passing, i.e., blind message and uniform passing, are revealed. These two simplifications cause the representations of all nodes be too similar to be distinguished from each other, i.e., over-smoothing issue. To overcome these two limits, a novel Diverse and Interactive Message Passing (DIMP) for self-supervised learning is proposed by implementing the message as the element-wise product of the representations of two connected nodes. It not only makes different neighbourhoods obtain different messages, but also induces different weights for different channels. The potential of DIMP on preventing over-smoothing issue is theoretically analyzed by proving it being a numerical optimization of overlapping community detection, and is experimentally evaluated on node-level and graph-level tasks.

Acknowledgements

This work was supported in part by the National Key R&D Program of China under Grant 2020YFB1406704, the National Natural Science Foundation of China under Grant 61972442, Grant 61802391, Grant 62102413, Grant U2001202, Grant U1936208, Grant 61876128 and Grant 61802282, in part by the Key Research and Development Project of Hebei Province of China under Grant 20350802D and 20310802D; in part by the Natural Science Foundation of Hebei Province of China under Grant F2020202040, in part by the Natural Science Foundation of Tianjin of China under Grant 20JCYBJC00650, in part by the China Postdoctoral Science Foundation under Grant 2021M703472, and in part by State Key Laboratory of Software Development Environment under Grant SKLSDE-2020ZX-18.

References

Adhikari, B.; Zhang, Y.; Ramakrishnan, N.; and Prakash, B. A. 2018. Sub2Vec: Feature Learning for Subgraphs. In *PAKDD*, 170–182.

Ball, B.; Karrer, B.; and Newman, M. E. J. 2011. Efficient and principled method for detecting communities in networks. *Phys. Rev. E*, 84: 036103.

Bo, D.; Wang, X.; Shi, C.; and Shen, H. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. 3950–3957.

Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020. Simple and Deep Graph Convolutional Networks. In *ICML*, 1725–1735.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*.

Chung, F. R.; and Graham, F. C. 1997. *Spectral graph theory*. 92. American Mathematical Soc.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*, 3837–3845.

Feng, W.; Zhang, J.; Dong, Y.; Han, Y.; Luan, H.; Xu, Q.; Yang, Q.; Kharlamov, E.; and Tang, J. 2020. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. In *NeurIPS*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*, 1263–1272.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*, 1024–1034.

Hassani, K.; and Ahmadi, A. H. K. 2020. Contrastive Multi-View Representation Learning on Graphs. In *ICML*, volume 119, 4116–4126. PMLR.

Jiao, Y.; Xiong, Y.; Zhang, J.; Zhang, Y.; Zhang, T.; and Zhu, Y. 2020. Sub-graph Contrast for Scalable Self-Supervised Graph Representation Learning. In *ICDM*, 222–231. IEEE.

Kipf, T. N.; and Welling, M. 2016. Variational Graph Auto-Encoders. *CoRR*, abs/1611.07308.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.

Li, Q.; Han, Z.; and Wu, X. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*, 3538–3545.

Ma, Y.; Liu, X.; Zhao, T.; Liu, Y.; Tang, J.; and Shah, N. 2020. A Unified View on Graph Neural Networks as Graph Signal Denoising. arXiv:2010.01777.

Monti, F.; Boscaini, D.; Masci, J.; Rodolà, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *CVPR*, 5425–5434.

Moon, T. K. 1996. The expectation-maximization algorithm. *IEEE Signal Process. Mag.*, 13(6): 47–60.

Namata, G.; London, B.; Getoor, L.; and Huang, B. 2012. Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*.

Narayanan, A.; Chandramohan, M.; Venkatesan, R.; Chen, L.; Liu, Y.; and Jaiswal, S. 2017. graph2vec: Learning Distributed Representations of Graphs. arXiv:1707.05005.

Oono, K.; and Suzuki, T. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *ICLR*.

Pan, S.; Hu, R.; Long, G.; Jiang, J.; Yao, L.; and Zhang, C. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding. In *IJCAI*, 2609–2615. ijcai.org.

Park, J.; Lee, M.; Chang, H. J.; Lee, K.; and Choi, J. Y. 2019. Symmetric Graph Convolutional Autoencoder for Unsupervised Graph Representation Learning. In *ICCV*, 6518–6527. IEEE.

Peng, Z.; Huang, W.; Luo, M.; Zheng, Q.; Rong, Y.; Xu, T.; and Huang, J. 2020. Graph Representation Learning via Graphical Mutual Information Maximization. In *WWW*, 259–270. ACM / IW3C2.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. DeepWalk: online learning of social representations. In *SIGKDD*, 701–710.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *ICLR*.

Sankar, A.; Wang, J.; Krishnan, A.; and Sundaram, H. 2020. Beyond Localized Graph Neural Networks: An Attributed Motif Regularization Framework. In *ICDM*, 472–481. IEEE.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2019. Pitfalls of Graph Neural Network Evaluation. arXiv:1811.05868.

Sun, F.; Hoffmann, J.; Verma, V.; and Tang, J. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.

Tao, Z.; Li, Y.; Ding, B.; Zhang, C.; Zhou, J.; and Fu, Y. 2020. Learning to Mutate with Hypergradient Guided Population. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Velickovic, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *ICLR*. OpenReview.net.

Wang, C.; Pan, S.; Long, G.; Zhu, X.; and Jiang, J. 2017. MGAE: Marginalized Graph Autoencoder for Graph Clustering. In *CIKM*, 889–898.

Wu, F.; Jr., A. H. S.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019. Simplifying Graph Convolutional Networks. In *ICML*, 6861–6871.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2021. A Comprehensive Survey on Graph Neural Networks. *TNNLS*, 32(1): 4–24.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.; and Jegelka, S. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*, 5449–5458.

Yanardag, P.; and Vishwanathan, S. V. N. 2015. Deep Graph Kernels. In *SIGKDD*, 1365–1374.

Yang, J.; and Leskovec, J. 2013. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM*, 587–596.

Yang, L.; Wang, C.; Gu, J.; Cao, X.; and Niu, B. 2021. Why Do Attributes Propagate in Graph Convolutional Neural Networks? In *AAAI*, 4590–4598.

Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *ICML*, 40–48.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph Contrastive Learning with Augmentations. In *NeurIPS*.

Zhao, L.; and Akoglu, L. 2020. PairNorm: Tackling Oversmoothing in GNNs. In *ICLR*.

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1: 57–81.

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D. 2020a. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *NeurIPS*.

Zhu, M.; Wang, X.; Shi, C.; Ji, H.; and Cui, P. 2021a. Interpreting and Unifying Graph Neural Networks with An Optimization Framework. In *WWW*, 1215–1226.

Zhu, R.; Tao, Z.; Li, Y.; and Li, S. 2021b. Automated Graph Learning via Population Based Self-Tuning GCN. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, 2096–2100. ACM.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2020b. Deep Graph Contrastive Representation Learning. *CoRR*, abs/2006.04131.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2021c. Graph Contrastive Learning with Adaptive Augmentation. In *WWW*. ACM. ISBN 9781450383127.