Motif Graph Neural Network

Xuexin Chen, Ruichu Cai*, Yuan Fang*, Min Wu*, Zijian Li, Zhifeng Hao

***Abstract*—Graphs can model complicated interactions between entities, which naturally emerge in many important applications. These applications can often be cast into standard graph learning tasks, in which a crucial step is to learn low-dimensional graph representations. Graph neural networks (GNNs) are currently the most popular model in graph embedding approaches. However, standard GNNs in the neighborhood aggregation paradigm suffer from limited discriminative power in distinguishing *high-order* graph structures as opposed to *low-order* structures. To capture high-order structures, researchers have resorted to motifs and developed motif-based GNNs. However, existing motif-based GNNs still often suffer from less discriminative power on high-order structures. To overcome the above limitations, we propose Motif Graph Neural Network (MGNN), a novel framework to better capture high-order structures, hinging on our proposed motif redundancy minimization operator and injective motif combination. First, MGNN produces a set of node representations w.r.t. each motif. The next phase is our proposed redundancy minimization among motifs which compares the motifs with each other and distills the features unique to each motif. Finally, MGNN performs the updating of node representations by combining multiple representations from different motifs. In particular, to enhance the discriminative power, MGNN utilizes an injective function to combine the representations w.r.t. different motifs. We further show that our proposed architecture increases the expressive power of GNNs with a theoretical analysis. We demonstrate that MGNN outperforms state-of-the-art methods on seven public benchmarks on both node classification and graph classification tasks.**

***Index Terms*—Graph Neural Network, Motif, High-order Structure, Graph Representation**

I. Introduction

Graphs are capable of modeling complex interactions between entities, which naturally emerge in many real-world scenarios. Social networks, protein-protein interaction networks, and knowledge graphs are just a few examples, with many important applications in areas like social recommendation [1], drug discovery [2], fraud detection [3], and particle physics [4]. These applications can often be cast into standard graph learning tasks such as node classification, link prediction, and graph classification, in which a crucial step is to learn low-dimensional graph representations.

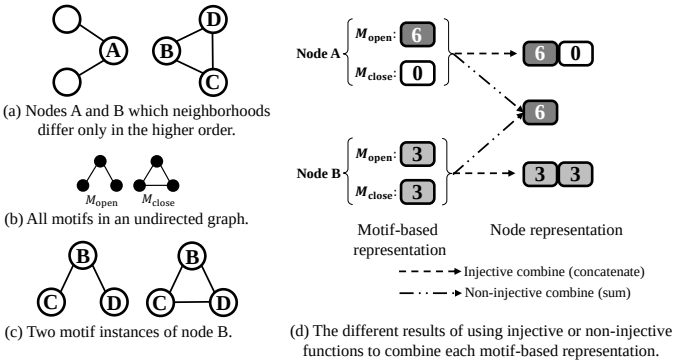

Fig. 1. Toy example of the discriminative power of GNNs on two nodes A and B with non-isomorphic neighborhoods.

Graph embedding approaches can be broadly categorized into graph neural networks (GNNs) [5], [6], [7], matrix factorization [8], [9] and skip-gram models [10], [11]. Among these, GNNs are currently the most popular model, largely owing to their ability of integrating both structure and content information through a message passing mechanism. To be more specific, in the standard GNN architecture, the representation vector of a node is computed by aggregating and updating messages (i.e., features or representation vectors) from the node's neighbors. The aggregation can be performed recursively by stacking multiple layers, to capture long-range node dependencies.

However, standard GNNs in the neighborhood aggregation paradigm suffer from limited discriminative power in distinguishing *high-order* graph structures consisting of the connections between neighbors of a node, as opposed to *low-order* structures consisting of the connections between the node and its neighbors. For example, standard GNNs cannot distinguish between nodes A and B with non-isomorphic neighborhoods in Fig. 1(a), as their neighborhoods differ only in the higher-order. To capture high-order structures, researchers have resorted to motifs [12], [13] and developed motif-based GNNs [14], [15], [16], [17]. These approaches usually employ a motif-based adjacency matrix for each motif, which is constructed from the number of times two nodes are connected via an instance of the motif. Such motif-based adjacency matrices can better grasp the high-order structures. For example, given the open and closed motifs illustrated in Fig. 1(b), nodes A and B in Fig. 1(a) can be naturally distinguished by motif-based GNNs since node A is only

This research was supported in part by National Key R&D Program of China (2021ZD0111501), National Science Fund for Excellent Young Scholars (62122022), Natural Science Foundation of China (61876043, 61976052), the major key project of Peng Cheng Lab (PCL2021A12), Guangdong Provincial Science and Technology Innovation Strategy Fund (2019B121203012).

Xuexin Chen is with the School of Computer Science, Guangdong University of Technology, Guangzhou 510006, China. E-mail: im.chenxuexin@gmail.com

Ruichu Cai is with the School of Computer Science, Guangdong University of Technology, Guangdong Provincial Key Laboratory of Public Finance and Taxation with Big Data Application, Guangzhou, China and also with Peng Cheng Laboratory, Shenzhen 518066, China. E-mail: cairuichu@gmail.com

Yuan Fang is with the School of Computing and Information Systems, Singapore Management University, 178902, Singapore. E-mail: yfang@smu.edu.sg

Min Wu is with the Institute for Infocomm Research (I^2R), A*STAR, 138632, Singapore. E-mail: wumin@i2r.a-star.edu.sg

Zijian Li is with the School of Computer Science, Guangdong University of Technology, Guangzhou 510006, China. E-mail: leizigin@gmail.com

Zhifeng Hao is with the College of Science, Shantou University, Shantou 515063, China. Email: haozhifeng@stu.edu.cn

associated with an open motif, whereas node B is associated with both open and close motifs.

However, existing motif-based GNNs often suffer from two problems. First, they overlook the *redundancy* among motifs, which is defined as common edges shared by different motif instances. For example, in Fig. 1(c), the two motif instances of node B share two edges. When the redundancy is high enough, different motifs may become similar and lack specificity. Second, they often combine multiple motifs in a *non-injective* manner, potentially resulting in less discriminative power on high-order structures. That is, a non-injective function, such as sum or mean, is used to combine different motifs, as shown in Fig. 1(d). In our example, node A has only an open motif with a feature valued 6, and node B has both open and close motifs each with a feature valued 3. However, when the motifs are combined by summing up their features, both nodes A and B would obtain the same feature representation of 6 and thus cannot be distinguished. Thus, the resulting node representations may also converge and further decrease the discriminative power.

To overcome the above limitations, we propose Motif Graph Neural Network (MGNN), a new class of GNN capable of distinguishing high-order structures with provably better discriminative power. From a model perspective, our MGNN follows the message passing mechanism, and its procedure is broken down into the following four phases. The first phase is motif instance counting. To form a motif-based adjacency matrix, we count the number of times two nodes co-occur in an instance of each motif. To capture comprehensive high-order graph structures, MGNN employs the motif-based adjacency matrices for all the possible motifs of size three, as opposed to some previous work with only one [18] or some motifs [17]. The second phase is message aggregation. MGNN, like other motif-based GNNs, aggregates node features (i.e., messages) on each motif-based adjacency matrix to produce different representations of the motifs. The first two phases are largely based on previous studies, except that we have employed all the motifs of size three to thoroughly capture high-order structures in an efficient manner. The third phase is the redundancy minimization among motifs. We address the challenge of redundancy among motifs by a proposed redundancy minimization operator, which compares the motifs with each other in terms of their representations, to distill the features specific to each motif. The fourth phase is the updating of node representations by combining multiple motifs. To improve the limited discriminative power of non-injective combinations, MGNN utilizes an injective function to combine motifs and update node representations. For example, to distinguish nodes A and B in Fig. 1(d), MGNN uses the injective concatenation to combine motif-based representations, so that the representation of node A is (6,0) and that of B is (3, 3), which can be differentiated apart. From a theoretical perspective, we show that MGNN is provably more expressive than standard GNN, and standard GNN is in fact a special case of MGNN.

We summarize our key contributions in the following.

- We propose Motif Graph Neural Network (MGNN), a novel framework to better capture high-order structures, hinging on the motif redundancy minimization operator and injective motif combination.
- We further show that our proposed architecture increases the expressive power of GNNs with a theoretical analysis.
- We demonstrate that MGNN outperforms existing standard or motif-based GNNs on seven public benchmarks on both node classification and graph classification tasks.

II. Related Work

Standard Graph Neural Networks follow the message passing paradigm to leverage node dependence and learn node representations. Different GNN models resort to different aggregation functions to aggregate the messages (i.e., features) for each node from its neighbors, and update its representation [5], [6], [7], [19], [20], [21], [22], [23], [24]. For example, Graph Convolutional Networks [5] use mean aggregation to pool neighborhood information. Graph Attention Networks [6] aggregate neighborhood information with trainable attention weights. GraphSAGE [7] uses mean, max or other learnable pooling function. Moreover, during aggregation, Message Passing Neural Networks [19] also incorporate edge information, while Graph Networks [20] and multi graph fusion-based dynamic GCN [21] further consider global graph information. In [22] and [23], GNNs are developed to use an aggregation strategy based on Hilbert-Schmidt independence criterion and self-paced label augmentation strategy, respectively. Some graph-level downstream tasks, such as graph classification, further employ a readout function to aggregate individual node representations into a whole-graph representation. The readout can be a simple permutation invariant function such as average and summation, while more sophisticated graph pooling methods have also been proposed, including global pooling [25], [26], [27], hierarchical pooling [28], [29], [30], [31], [32]. Besides graph-level downstream tasks, GNNs can also be used for the visual question answering task, etc [24]. However, all these models are limited to only capturing low-order graph structures around every node.

However, standard GNNs are at most as powerful as the 1-dimensional Weisfeiler-Leman (1-WL) graph isomorphism test [33], which implies that they cannot distinguish nodes with isomorphic low-order graph structures but different high-order structures. In other words, standard GNNs will always associate such nodes with the same representation. To improve the discriminative power of GNNs, it is a common practice to leverage high-order graph structures such as motifs [12], [13]. In particular, motif-based GNN models use one [18], [15], [34] or more [16], [17], [35], [36], [14], [37], [38] motif-based adjacency matrices to perform message passing. When multiple motif-based adjacency matrices are used, the combination function w.r.t. multiple motifs include summation [16], [37], averaging [17], neighborhood aggregation [35], fusion by a fully connected layer [36], selection by reinforcement learning [14], combination by a variant of recurrent neural network [38], and so on. However, all these previous combination functions are not injective to sufficiently differentiate higher-order structures. Note that although the model in [38] does not employ an injective function, it still effectively captures

the high-order structure of the nodes, through a strategy of encoding neighbor's features sequentially and a variant of the recurrent neural network to learn the node representations. Besides, all these models do not take into account the redundancy among motif instances.

In another line, several studies [39], [40], [41] attempt to extend the discriminative power of GNNs from 1-WL to k-WL, given that the higher the dimension of WL, the stronger the discriminative power. Like the standard GNNs, there is also a message propagation mechanism in these k-WL approaches where the difference is that the message is not propagated between nodes but between k-tuples (or a subgraph with k nodes). Since their message propagation is not between nodes, they [39], [40], [41] have the following shortcomings compared with our MGNN. First, they cannot generate node embeddings, but MGNN can, which limits their application to node-level tasks such as node classification. Second, their time complexity is higher than that of MGNN. The time complexity of MGNN is $O(|V|^2)$ (see Section IV-F), while their time complexity is $O(|V|^3)$ [39] or even $O(|V|^4)$ in the worst case [40], [41]. Third, [40], [41] have a space complexity $O(|V|^3)$, which is also higher than $O(|V|^2)$ needed by MGNN.

There are also several approaches employing high-order structures, in which each node receives messages from its multi-hop neighbors, such as MixHop [42], GDC [43], CADNet [44], PathGCN [45], SE-aggregation [46] and MBRec [47]. However, like standard GNNs, they are typically at most as powerful as the 1-WL test in distinguishing graph structures [46].

III. Preliminaries

In this section, we introduce major notations and definitions of related concepts.

A. Notations and Problem Formulation

A graph is denoted by $G = (V, E)$, with the set of nodes V and the set of edges E. Let $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$ be the adjacency matrix of G, and $\mathbf{X} \in \mathbb{R}^{|V| \times d_0}$ be the node feature matrix of G, where i-th row means the features of node i denoted by $\mathbf{x}_i$. We use $(\mathbf{T})_{ij}$ to represent the element in the i-th row and the j-th column of a matrix $\mathbf{T}$, and $(\mathbf{T})_i$ to represent all of the i-th row's elements.

In this paper, we investigate the problem of graph representation learning, which aims to embed nodes into a low-dimensional space. The node embeddings can be used for downstream tasks such as node classification, potentially in an end-to-end fashion. Formally, a node embedding model is denoted by a function $: V \to H$ that maps the nodes in V to d-dimensional vectors in $H = \{\mathbf{h}_i \in \mathbb{R}^d | 1 \le i \le |V|\}$, where i denotes the index of the nodes.

B. Motif and Motif-based Adjacency Matrix

We work with directed motifs because they allow us to describe more complex structures. Specifically, we first introduce the definition of motif [12], [18], [13] as follows.

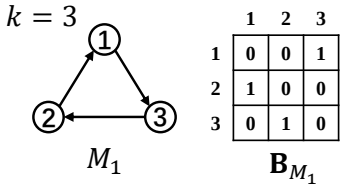

Fig. 2. An example of a 3-node ($k = 3$) network motif, along with its adjacency matrix $\mathbf{B}_{M_1}$.

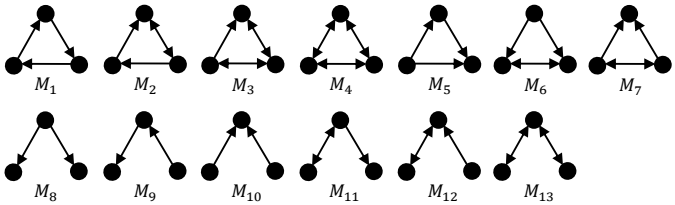

Fig. 3. All 3-node motifs in a directed and unweighted graph.

Definition 1. *(Network motif). A motif M is a connected graph of n nodes ($n > 1$), with a $n \times n$ adjacency matrix $\mathbf{B}_M$ containing binary elements $\{0, 1\}$.*

An example of 3-node motif is given in Fig. 2. In particular, a motif with three or more nodes (i.e., $n \ge 3$) can capture high-order graph structures. Among them, w.r.t. a given node, the high-order structure captured by its motifs with $n > 3$ nodes (i.e., not only its edges incident to its neighboring nodes but also the edges between its neighboring nodes), can be similarly captured by multiple 3-node motifs. Thus, the given node's 3-node motifs have sufficient capacity for structures. As shown in Fig. 3, we enumerate a total of thirteen 3-node motifs. Therefore, we only utilize motifs with $n = 3$ nodes in this work.

Given the above motif definition, we can further define the set of motif instances as follows.

Definition 2. *(Motif instance). Consider an edge set E^0 and the subgraph $G[E^0]$ induced from E^0 in G. If $G[E^0]$ and a motif M_k are isomorphic [48], written as $M_k \simeq G[E^0]$, then*

$$m(E^0) = \{(\mathbf{x}_u, \mathbf{x}_v) \mid (u, v) \in E^0\}$$

is an instance *of the motif M_k, where u, v are two adjacent nodes that form an edge in E^0, and $\mathbf{x}_u$ means the u-th row of $\mathbf{X}$ (i.e., the feature vector of node u).*

For example, a motif instance of M_1 in Fig. 2 is $\{(\mathbf{x}_1, \mathbf{x}_3), (\mathbf{x}_2, \mathbf{x}_1), (\mathbf{x}_3, \mathbf{x}_2)\}$.

Definition 3. *(Motif instance set). On a graph $G = (V, E)$, the set of instances of motif M_k, denoted as $\mathcal{M}_k$, is defined by*

$$\mathcal{M}_k = \{m(E^0) \mid E^0 \subseteq E, |E^0| = r, M_k \simeq G[E^0]\},$$

where $E^0 \subseteq E, |E^0| = r$ denotes the set of all r-combinations of the edge set E, and $|E^0| = r$ is the number of edges in the motif M_k.

Based on the motif instances, the definition of the motif-based adjacency matrix is given as follows.

Definition 4. *(Motif-based adjacency matrix). Given a motif M_k and its set of instances $\mathcal{M}_k$, the corresponding motif-based adjacency matrix $\mathbf{A}_k$ is defined by*

$$(\mathbf{A}_k)_{ij} = \sum_{m \in \mathcal{M}_k} \mathbb{I}((x_i, x_j) \in m), \quad (1)$$

where $\mathbb{I}(\cdot)$ is an indicator function, i.e., $\mathbb{I}(x) = 1$ if the statement x is true and 0 otherwise.

Intuitively, $(\mathbf{A}_k)_{ij}$ is the number of times two nodes i and j are connected via an instance of the motif M_k.

IV. Proposed Approach

In this section, we introduce the proposed approach. We first present an overall framework of our approach, followed by its four phases in detail. Finally, we discuss the overall objective function for model training.

A. Overall Framework

We propose Motif Graph Neural Network (MGNN) that can model high-order structures with provably better discriminative power. Specifically, our MGNN follows a message passing mechanism, and its procedure is broken down into the following four phases.

The first phase involves the construction of a motif-based adjacency matrix, as shown in Fig. 4(a). Given a motif, its motif-based adjacency matrix captures the number of times each pair of nodes are connected via an instance of the motif. Thus, we need an efficient counting algorithm for motif instances. In MGNN, we consider all 13 motifs of size three, namely $M_1, M_2, \ldots, M_{13}$ given by Fig. 3, and correspondingly construct 13 motif-based adjacency matrices $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_{13}$. The second phase is message aggregation, as shown in Fig. 4(b). MGNN aggregates node features (i.e., messages) on each motif-based adjacency matrix to produce a set of node representations w.r.t. each motif. The first two phases of motif instance counting [18] and message aggregation [16], [36] are largely similar to previous works, except that we have employed all the motifs of size three to comprehensively capture high-order structures in an efficient manner. In the second phase, we follow previous work completely. The third phase is the redundancy minimization among motifs, as shown in Fig. 4(c). We propose a redundancy minimization operator, which compares the motifs with each other and distills the features unique to each motif. The fourth phase performs the updating of node representations by combining multiple representations from different motifs, as shown in Fig. 4(d). In particular, to enhance the discriminative power, MGNN utilizes an injective function to combine the representations w.r.t. different motifs.

B. Motif-based Adjacency Matrix Construction

The key step to constructing a motif-based adjacency matrix is to efficiently count the number of motif instances. Depending on if the motif is open (M_8–M_{13}) or closed (M_1–M_7), different counting algorithms will apply.

For open motifs (M_8–M_{13}), existing methods [49], [50] are often implemented by enumerating motif instances. For example, given the graph in Fig. 5(a), to construct the M_9-based adjacency matrix, a traditional technique is to enumerate the instances of M_9 as shown in Fig. 5(b). However, such enumeration suffers from high computational complexity, with a worst-case complexity of $O(|V|^3)$ in both space and time. To reduce the complexity, we propose an adjacency matrix construction method for open motifs without enumerating any motif instance, which has a time and space complexity of $O(|V|^2)$ and $O(|V|)$, respectively. Consider a node v. Let u_{in}, u_{out}, and u_{bi} denote an incoming, outgoing, and bi-directional neighbor of node v, respectively. Correspondingly, let d_{in}, d_{out} and d_{bi} denote the number of each type of neighbor of v, respectively, as illustrated by the examples in Fig. 5(a). As shown in Fig. 3, the center node of each open motif has at most two types of neighbors; for example, M_9 has u_{out} and u_{in}, and M_{13} has only u_{bi}. Our key observation is that $(\mathbf{A}_k)_{vu}$, the number of times two nodes (v and u) are connected via an instance of an open motif M_k, can be computed as follows. On one hand, when the motif has two types of neighbors, $(\mathbf{A}_k)_{vu}$ will be equal to the number of the other type of neighbors, e.g., $(\mathbf{A}_9)_{vu_{\text{in}}} = d_{\text{out}}$, $(\mathbf{A}_9)_{vu_{\text{out}}} = d_{\text{in}}$, $(\mathbf{A}_{11})_{vu_{\text{out}}} = d_{\text{bi}}$, $(\mathbf{A}_{12})_{vu_{\text{in}}} = d_{\text{bi}}$, $(\mathbf{A}_{11})_{vu_{\text{bi}}} = d_{\text{out}} - 1$, $(\mathbf{A}_{12})_{vu_{\text{bi}}} = d_{\text{in}} - 1$. On the other hand, when the motif has only one type of neighbors, $(\mathbf{A}_k)_{vu}$ will be equal to the number of neighbors in the motif, e.g., $(\mathbf{A}_8)_{vu_{\text{out}}} = d_{\text{out}} - 1$, $(\mathbf{A}_{10})_{vu_{\text{in}}} = d_{\text{in}} - 1$, $(\mathbf{A}_{13})_{vu_{\text{bi}}} = d_{\text{bi}} - 1$. Still using Fig. 5 as an example, node B is an incoming neighbor (u_{in}) of node A, while C and D are the outgoing neighbors (u_{out}) of node A. Furthermore, for $(\mathbf{A}_9)_{AB}$, it satisfies $(\mathbf{A}_9)_{vu_{\text{in}}} = d_{\text{out}}$ (denoting node A as v) and for $(\mathbf{A}_9)_{AC}$ or $(\mathbf{A}_9)_{AD}$, it satisfies $(\mathbf{A}_9)_{vu_{\text{out}}} = d_{\text{in}}$.

The motif-based adjacency matrix for a closed motif (M_1–M_7) can be constructed by an existing method [18] with a time and space complexity of $O(|V|^3)$ and $O(|V|^2)$, respectively, which counts $(\mathbf{A}_k)_{vu}$ through two matrix multiplication operations and the matrices used by this method can be stored in the HDF5 format [51].

C. Motif-wise Message Aggregation

To produce the motif-wise node representations, on each motif-based adjacency matrix, node features (i.e, messages) can be incorporated into a multi-layer message aggregation mechanism, as shown in Fig. 4(b).

Specifically, the motif M_k-based representation of node v in the l-th layer is given by

$$\mathbf{h}_{v;k}^{(l)} = \text{AGG}\left(\left\{ (\hat{\mathbf{A}}_k)_{vi} (\mathbf{Z}^{(l)})_i \mid i \in \mathcal{N}(v) \right\}\right), \quad (2)$$

$$\mathbf{Z}^{(l)} = \hat{\mathbf{A}} \mathbf{H}^{(l-1)} \mathbf{W}^{(l)}, \quad (3)$$

where $\mathbf{H}^{(l-1)} \in \mathbb{R}^{|V| \times d_{l-1}}$ denote the node messages from the previous $(l-1)$-th layer and $\mathbf{H}^{(0)} = \mathbf{X}$, $\mathbf{W}^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$ is the trainable weight matrix in the l-th layer. $\hat{\mathbf{A}}$ is the normalized adjacency matrix given by $\hat{\mathbf{A}} = \hat{\mathbf{A}} - \frac{\hat{\lambda}_{\max}}{2}\mathbf{I}$, where $\hat{\mathbf{A}} = \mathbf{D}^{-0.5}\mathbf{A}\mathbf{D}^{-0.5}$ and $\mathbf{D}$ is a diagonal matrix in which the diagonal elements are defined as $(\mathbf{D})_{ii} = \sum_{j=1}^{|V|} (\mathbf{A})_{ij}$ and

Fig. 4. Overview of a MGNN layer. The MGNN layer takes the graph and its node features matrix as inputs, throughout the procedure (a) (b) (c) (d), and finally outputs the node representation matrix that captures the high-order graph structure. The first two phases are the generation of motif- based representation of nodes. The key idea of phase (c) is to compare the motifs with each other and distill the features specific to each motif. In phase (d), motif- updated representations are combined by an injective concatenation operation.

Fig. 5. Overview for constructing M_9-based adjacency matrix by enumeration and non-enumeration method.

$\hat{\lambda}_{max}$ refers to the largest eigenvalue of $\hat{A}$. The above normalization technique aids in the centralization of the Laplacian's eigenvalues and the reduction of the spectral radius bound [52]. The motif M_k-based adjacency matrix A_k is normalized in the same way into $\tilde{A}_k$. AGG, the function of $H^{(l-1)}$ and $\tilde{A}_k$ as Fig. 4(b) illustrated, is a message aggregate function, e.g., sum, mean or max. The coefficient $\pi^{(l)}_{k;vi}$ is the attention weight that indicates the importance of node i's messages to node v. $\pi^{(l)}_{k;vi}$ can be assigned a constant value according to prior knowledge or computed by the attention mechanism [6]. $\mathcal{N}(v)$ represents the set of neighboring nodes of v. Note that not all nodes will have 13 motifs, and MGNN can still accommodate such nodes. In particular, if node v lacks a motif M_k, the entries $(A_k)_{vj}$ and $(A_k)_{jv}$ in Eq. (1) are all set to zeros, and subsequently $h^{(l)}_{v;k}$ in Eq. (2) will also be a zero vector.

Intuitively, in Eq. (2), before performing motif-wise aggregation for the motif M_k, we first stack a GCN layer [5], i.e., $Z = \tilde{A}H^{(l-1)}W^{(l)}$ in Eq. (3), to update the overall node messages by aggregating from the previous layer. The GCN layer can also be replaced by other GNN layers.

D. Motif Redundancy Minimization

As different motifs often share certain substructures, the corresponding motif-wise representations may become similar and lack specificity. Inspired by the idea of redundancy minimization between features [53], we propose a redundancy minimization operator at the motif level, denoted $\ominus$. The key idea of $\ominus$ is to compare the motifs with each other and adaptively distill the features specific to each motif. We formally define $\ominus$ as follows. Given a node v, for simplicity, let h_k and z_v denote $h^{(l)}_{v;k}$, $(Z^{(l)})_v$ respectively. We call the motif- and GCN-based representations collectively as the intermediate representations of the node.

Definition 5. (Motif redundancy minimization operator). For any node v, given its intermediate representations h_1, ..., $h_{13}; z_v$, let $H_k = \Vert_{i=1;i\neq k}^{13} h_i \Vert z_v$, where $\Vert$ is the concatenation operator. In other words, H_k concatenates all the intermediate representations except that based on motif M_k. Then, for motif M_k, its redundancy minimized representation of the node v is given by

$$\begin{aligned} \tilde{h}_k &= \ominus(\alpha_k; h_1; \cdots; h_{13}; z_v) \\ &= \alpha_k \, f(h_k) \odot f_k(H_k) \, : \end{aligned} \tag{4}$$

$\tilde{h}_k$ is the updated representation of h_k after redundancy minimization. $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ is a learnable projection function to map the intermediate motif-based representations to the same space as its redundant features. And $f_k : \mathbb{R}^{13d} \to \mathbb{R}^{d'}$ is a learnable feature selection function, which selects the redundant features w.r.t. motif M_k. α_k is the similarity between $f(h_k)$ and $f_k(H_k)$, which acts as a regularizer to prevent extremely small or large differences between the two terms. σ is an activation function (e.g., ReLU).

Intuitively, in Eq. (4), the motif redundancy minimization operator subtracts or removes redundant features w.r.t. each motif from the intermediate representations of a given node. Apart from minimizing the redundancy, the operator also performs an adaptive selection of motifs in general. That is, for an unimportant motif M_k, this operator will make $\tilde{h}_k$ in Eq. (4) close to a zero vector through functions α_k and f_k. In particular, when $\tilde{h}_k$ is a zero vector, it is equivalent to removing the instance of M_k containing node v in Eq. (1). In Section VI-C, we will use a heatmap to demonstrate this adaptive selection mechanism, which improves the robustness of MGNN.

To realize the motif redundancy minimization operator, we need to instantiate f, f_k and $_k$ in Eq. (4). In particular, we use a fully connected layer to f and f_k, namely,

$$f(h_k) = W_f^{(l)} h_k + b_f^{(l)}; \quad (5)$$

$$f_k(H_k) = W_{f_k}^{(l)} H_k + b_{f_k}^{(l)}; \quad (6)$$

where $W_f^{(l)} \in R^{d_l^0 \times d_l}$ is a trainable matrix in the l-th layer shared by all motifs, and $W_{f_k}^{(l)} \in R^{d_l^0 \times (13 d_l)}$ is a trainable matrix specific to motif M_k in the l-th layer, and $b_f^{(l)} \in R^{d_l^0}$ and $b_{f_k}^{(l)} \in R^{d_l^0}$ are the corresponding bias vectors. Furthermore, to measure the similarity $_k$, we use the inner product with a non-linear activation (e.g., sigmoid or tanh), that is,

$$_k = \quad f(h_k)^{\top} f_k(H_k) : \quad (7)$$

in Eq.(4) and Eq.(7) might be the same or distinct.

Through the above instantiations, we can minimize the redundancy among motifs as shown in Fig. 4(c), for every node in every layer. That is,

$$\tilde{h}_{v;k}^{(l)} = \quad k; h_{v;1}^{(l)}; :::; h_{v;13}^{(l)}; (Z^{(l)})_v \quad ; \quad (8)$$

where $\tilde{h}_{v;k}^{(l)} \in R^{d_l^0}$ is the updated representation of node v based on motif M_k in the l-th layer.

E. Node Representation Update via Injective Function

As shown in Fig. 4(d), MGNN updates the node representation by combining their intermediate, motif-based representations. To improve the discriminative power on high-order structures, MGNN utilizes an injective function to combine different motif-based representations of each node, to update the output node representations in each layer. Specifically, we use the injective vector concatenation function, and generate the output node representation in the l-th layer below.

$$h_v^{(l)} = \quad _{k=1}^{13} \tilde{h}_{v;k}^{(l)}; \quad (9)$$

where $h_v^{(l)} \in R^{13 d_l}$ is the output representation of node v in the l-th layer.

The following two properties of the concatenation function are essential to increase the expressive power of MGNN. First, the output node representation $h_v^{(l)}$ will not change if the order of concatenation and aggregation is interchanged. Second, $h_v^{(l)}$ can always explicitly preserve each motif-based feature embedding via the injective combination. Using these two properties, we can theoretically show that MGNN has a larger representation capacity than the standard GNN, as we will further discuss in Section V.

F. Model Training

The node representations generated by MGNN can be used for various downstream learning tasks, including supervised and unsupervised learning.

For supervised learning, the node representations can be directly used as features for a specific downstream task, optimized with a supervised loss that can be abstracted as

$$L(Y; \hat{Y}); \quad (10)$$

$$\hat{Y} = (H^{(L)}); \quad (11)$$

where $\hat{Y}$ is the predicted matrix, $H^{(L)}$ is the node representation matrix generated by the last, L-th MGNN layer, such that its i-th row is the embedding vector $h_i^{(L)}$ of node i in Eq. (9). The loss function L, prediction function and the ground truth Y depend on the specific downstream task. Taking node classification as an example, the loss can be the cross-entropy loss over the training samples, as follows.

$$X \quad X^{n_c} \quad (Y)_{ij} \log(\hat{H}^{(l)})_{ij} ; \quad (12)$$

$$i 2 Y \; j=1$$

where Y is the set of training node indices, n_c denotes the number of classes, Y is the ground truth matrix such that its i-th row is the one-hot label vector of node i, and $\hat{H}^{(l)}$ is the predicted matrix such that its i-th row is the predicted class distribution of node i, which can be obtained by taking a softmax activation or additional neural network layers as the prediction function and passing $H^{(l)}$ through . Another common supervised task is graph classification, which can use a similar cross-entropy loss and prediction function, but the node representations must first undergo a readout operation [33] to generate the graph-level representations.

For unsupervised learning, the node representations can be trained through the graph auto-encoder [54] or other self-supervised frameworks [55] without any task-specific supervision.

Algorithm 1 summarizes the framework of MGNN. To be more specific, MGNN takes the node features, normalized adjacency matrix, and motif-based adjacency matrices for all possible motifs as inputs. The construction of motif-based adjacency matrices is based on our proposed method and another method in the literature mentioned in section IV-B. MGNN further propagates the node representations (or node input features) layer by layer. In each layer, first, from line 4 to line 7, MGNN produces the M_k-based representation $\tilde{h}_{v;k}^{(l)}$ of node v by performing message aggregation. Second, from line 8 to line 14, MGNN compares the M_k-based representation with each other using the motif redundancy minimization operator, to distill the features specific to each motif. Third, in line 15, MGNN utilizes the injective concatenation to combine M_k-based representations and update the representation of the node. Finally, in line 18, the set of output representations of each node is returned.

The computational complexity of one MGNN layer is $O(|V|^2)$, as follows. Firstly, the complexity is dominated by the computations in Eqs. (2) and (3), where the time complexities are given by $O(|V|d)$ and $O(|V|^2 d)$, respectively (d denotes the dimension of an MGNN layer). Hence, when computing Eq. (2) over all the nodes, the complexity is $O(|V|^2 d)$. Secondly, in our implementation, Eq. (3) can be pre-calculated before Eq. (2). Therefore, the overall time complexity of MGNN is $O(|V|^2 d)$, which can be further simplified to $O(|V|^2)$ as d is typically a small constant.

V. Theoretical Analysis

In this section, we aim to analyze the representation capacity of MGNN in comparison with standard GNN. In order to

Algorithm 1: The framework of MGNN.

Input: Node input feature matrix $X \in \mathbb{R}^{|V| \times d_0}$, normalized adjacency matrix $A \in \mathbb{R}^{|V| \times |V|}$, and normalized motif-based adjacency matrices $\tilde{A}_k \in \mathbb{R}^{|V| \times |V|}$, $k \in \{1, \ldots, 13\}$.
Output: Node embedding $h_v^{(L)} \in \mathbb{R}^{13 d_L}$ for each node.

```
Randomly initialize all parameters
H^(0) ← X
for l = 1, ..., L do
    for v ∈ V do
        for k = 1, ..., 13 do
            h_{v;k}^(l) ← Compute the M_k-based representation of node v by Eq. (2)
        end
        for k = 1, ..., 13 do
            f(h_{v;k}^(l)) ← Map h_{v;k}^(l) to the same space by Eq. (5)
            H_k ← Concatenate each h_{v;i}^(l) (i ≠ k) and z_v according de nition 5
            f_k(H_k) ← Select the redundant feature w.r.t. motif M_k by Eq. (6)
            α_k ← Compute the similarity between f(h_{v;k}^(l)) and f(H_k) by Eq. (7)
            h̃_{v;k}^(l) ← Update h_{v;k}^(l) based on f(h_{v;k}^(l)), f_k(H_k) and α_k by Eq. (8)
        end
        h_v^(l) ← Concatenate each h̃_{v;k}^(l) by Eq. (9)
    end
end
return {h_v^(L) | v ∈ V}
```

facilitate the analysis, we first introduce a simplified version of MGNN, and then further show that even the simplified MGNN still has stronger discriminative power than standard GNN.

A. Simplified Version of MGNN

A simplified version of the l-th MGNN layer is as follows:

$$h_{v;k}^{(l)} = \left\{ \sum_{i \in \mathcal{N}(v)} (\tilde{A}_k)_{vi} W_m^{(l)} h_i^{(l-1)} \right\}; \quad (13)$$

$$h_v^{(l)} = \Big\|_{k=1}^{13} (h_{v;k}^{(l)}); \quad (14)$$

where $\{\}$ represents the aggregate function.

Then we demonstrate that Eqs. (13)–(14) is a simplified version of a MGNN layer. Specifically, in Eq. (2), normalized $\tilde{A}_k$ and $(Z^{(l)})_i$ are substituted for A_k as well as $W_m^{(l)} h_i^{(l-1)}$, respectively, where $W_m^{(l)}$ is the trainable weight matrix in the l-th simplified MGNN layer and $h_i^{(l-1)}$ is the node messages from the previous $(l-1)$-th simplified MGNN layer ($h_j^{(0)} = x_i$). After that, Eq. (13) is obtained. Then, the output $h_{v;k}^{(l)}$ of Eq. (13) is utilized in place of $\tilde{h}_{v;k}^{(l)}$ in Eq. (9) and then Eq. (14) is obtained. Thus, Eqs. (13)–(14) is a simplified version of a MGNN layer.

TABLE I
THE LAYER OF THE ABSTRACT MODEL OF THE STANDARD GNN OR THE SIMPLIFIED VERSION OF MGNN.

Step1	message aggregation
Standard GNN	$h_v^{(l)} = \left\{ (A)_{vi} W_s^{(l)} h_i^{(l-1)} \mid i \in \mathcal{N}(v) \right\}$
MGNN	$h_{v;k}^{(l)} = \left\{ \sum_{i \in \mathcal{N}(v)} (\tilde{A}_k)_{vi} W_m^{(l)} h_i^{(l-1)} \right\}$
Step2	**node representation update**
Standard GNN	$\tilde{h}_v^{(l)} = (h_v^{(l)})$
MGNN	$h_v^{(l)} = \Big\|_{k=1}^{13} (h_{v;k}^{(l)})$

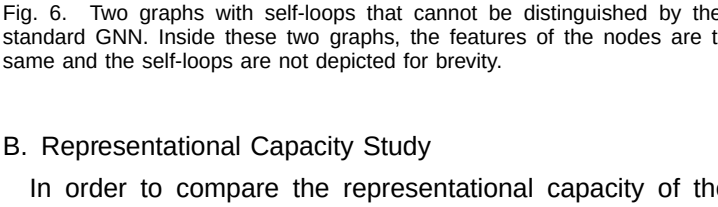

Fig. 6. Two graphs with self-loops that cannot be distinguished by the standard GNN. Inside these two graphs, the features of the nodes are the same and the self-loops are not depicted for brevity.

B. Representational Capacity Study

In order to compare the representational capacity of the simplified version of MGNN with that of the standard GNN, we begin with the layers of the abstract model of the standard GNN and the simplified version of MGNN in Table I, where $W_s^{(l)}$ is the trainable weight matrix in the l-th standard GNN layer, and $h_i^{(l-1)}$ is the node messages from the previous $(l-1)$-th standard GNN layer ($h_i^{(0)} = x_i$). The mainstream models of GNNs, including GCN, GAT, GraphSAGE and GIN, can be viewed as an instance of the standard GNN. Then we shows MGNN has larger representational capacity than standard GNN in Lemmas 1–2 and Theorem 1.

Based on the above abstractions, we first show that even a special case of MGNN at least has the same representational capacity as the standard GNN in Lemma 1.

Lemma 1. Given any an instance of the standard GNN, if the aggregate functions of standard GNN and MGNN are the same and the input to $\{\}$ only consist of values in the same dimension from different feature vectors, its representations of the graphs can also be generated by a special case of MGNN.

The proof for Lemma 1 is hinged on two important properties of the injective concatenation function, i.e., the interchangeability of concatenation and aggregation, and the explicit preservation of motif-based representations, as first mentioned in Section IV-E. The detailed proof can be found in Section I of our supplementary materials. In short, Lemma 1 shows that a standard GNN can be subsumed by MGNN. Taking one step further, we show that there exist two graphs that can be distinguished by MGNN but are indistinguishable by the standard GNN.

Lemma 2. There exist two non-isomorphic graphs G and G' with self-loops, which can be distinguished by MGNN, but not by the standard GNN.

Proof. As Fig. 6 illustrates, consider the two non-isomorphic

graphs G and G^0 with self-loops, in which all nodes have the same features. First, G and G^0 cannot be distinguished by standard GNN, because the multi-set of neighboring features of each node are the same. Second, G and G^0 can be naturally distinguished by MGNN since each node in G is only associated with an open motif, whereas each node in G^0 is associated with both open and close motifs. □

Based on the results on Lemma 1 and Lemma 2, we immediately come to the conclusion about the representation capacity in Theorem 1.

Theorem 1. MGNN has a larger representation capacity than the standard GNN.

VI. Experiments

In this section, we introduce the details of the experimental setup and the comparison results.

A. Experimental Setup

1) Datasets: To evaluate the effectiveness of our proposed MGNN, we utilize seven public datasets on two benchmark tasks: (1) classifying nodes on three citation network datasets (Cora, Citeseer, and Pubmed) and a knowledge graph (Chem2Bio2RDF), and (2) classifying graphs on three biochemical graph datasets (MUTAG, ENZYMES, and AIDS). Table II summarizes the statistics of seven datasets.

Cora, Citeseer and Pubmed [56] contain documents represented by nodes and citation links represented by edges.

Chem2Bio2RDF [57] integrates data from multiple public sources. Because the node feature is not provided in Chem2Bio2RDF and the discriminative power of GNN-based methods often depends on the properties of nodes, we use the degree statistical information of each node and its 1-hop neighborhood (5 dimensions in total) [58] as its node features.

MUTAG [59] contains 188 chemical compounds divided into two classes according to their mutagenic effect on a bacterium.

ENZYMES [60] contains 100 proteins from each of the 6 Enzyme Commission top level enzyme classes.

AIDS[1] [61] contains 2 classes (active, inactive), which represent molecules with activity against HIV or not.

2) Baselines: We consider three categories of methods, namely low- and high-order GNN-based methods, network embedding-based methods, as well as graph pooling-based methods. Low-order GNN-based methods include:

GCN [5] aggregates the feature information from a node's neighborhood.

GraphSAGE [7] generates embeddings by sampling and aggregating features from a node's local neighborhood.

GAT [6] incorporates the attention mechanism into the propagation step, following a self-attention strategy.

GIN [33] performs the feature aggregation in an injective manner based on the theory of the 1-WL graph isomorphism test.

BGNN [62] combines gradient boosted decision trees (GBDT) with GNN.

High-order GNN-based methods include:

MotifNet [16] utilizes a Laplacian matrix based on multiple motif-based adjacency matrices as the convolution kernel of the graph, and uses an attention mechanism to select node features.

MixHop [42] concatenates the aggregated node features from neighbors at different hops in each layer.

GDC [43] utilizes generalized graph diffusion (e.g. Personalized PageRank) to generate a new graph, then uses this new graph to predict rather than the original graph.

CADNet [44] obtains neighborhood representations by random walks with attention, and incorporates the neighborhood representations via trainable coefficients.

Network embedding-based methods include:

DeepWalk [11] combines truncated random-walk with skip-gram model to learn node embedding.

GraRep [9] leverages various powers of the adjacency matrix to capture higher-order node similarity.

HOPE [8] preserves higher-order proximity in node representations.

Node2Vec [10] employs biased-random walks, which provide a trade-off between breadth-first (BFS) and depth-first (DFS) graph searches, to learn node embedding.

Graph2Vec [63] creates WL tree for nodes as features in graphs to decompose the graph-feature co-occurence matrix.

NetLSD [64] calculates the heat kernel trace of the normalized Laplacian matrix over a vector of time scales.

GL2Vec [65] extends Graph2Vec with edge features by utilizing the line graph.

Feather [66] describes node neighborhoods with random walk weights.

Graph pooling-based methods include:

Graclus [28] is an alternative of eigen-decomposition to calculate a clustering version of the original graph.

GlobalATT [26] employs gate recurrent unit architectures with global attention to update node latent representations.

EdgePool [30] extracts graph features by contracting edges and merging the connected nodes uniformly.

TopKPool [31] learns a scalar projection score for each node and selects the top-k nodes.

ASAP [29] utilizes a self-attention network along with a modified GNN formulation to capture the importance of each node in a given graph.

We use the low- and high-order GNN-based approaches for both node classification and graph classification tasks, except that BGNN is used for the node classification task only since its GNN module is designed to provide the gradients generated by the node classification loss during training [62]. We use the following node-level network embedding

[1] https://wiki.nci.nih.gov/display/NCIDTPdata/AIDS+Antiviral+Screen+Data

TABLE II
STATISTICS OF THE DATASETS

Category	Dataset	# Graphs	# Nodes (Avg.)	# Edges (Avg.)	# Features	# Classes
Citation Graphs	Cora	1	2,708	5,429	1,433	7
	Citeseer	1	3,327	4,732	3,703	6
	Pubmed	1	19,717	44,338	500	3
Knowledge Graphs	Chem2Bio2RF	1	295,911	727,997	5	10
Biochemical Graphs	MUTAG	188	17.90	19.79	7	2
	ENZYMES	600	32.63	62.14	21	6
	AIDS	2,000	15.69	16.20	42	2

approaches for the node classification task only: DeepWalk [11], GraRep [9], HOPE [8], and Node2Vec [10]. We use all the graph pooling approaches and the following graph-level network embedding approaches for the graph classification task only: Graph2Vec [63], NetLSD [64], GL2Vec [65], and Feather [66].

3) Implementation details: The configurations of our MGNN as well as GNN-based baselines on the node classification task are as follows. We use 1 GNN layer for Cora and Citeseer datasets, while 2 GNN layers for the other two larger datasets namely Pubmed and Chem2Bio2RDF. In addition, a fully connected layer (FCL) is added after the last GNN layer to further process the node representation matrix. For the graph classification task, we use 3 GNN layers on all biochemical graph datasets for MGNN, GNN-based baselines as well as graph pooling-based baselines. Similarly, the node representation matrix after the last GNN layer would be passed through three fully connected layers. We use sum aggregation as the readout operation to derive the embedding for the graph.

For our MGNN, aggregate function AGG in Eq. (2) was sum. The activation function in Eq. (7) was set as sigmoid for Cora and Citeseer, while it was set as tanh for other datasets [6]. We further set $d_1; d_2; d_3$ in Eq. (2), the output dimensionality of the GCN layer which is stacked in the first, second, and third MGNN layers, to $16; n_c; n_c$, respectively, where n_c is the number of classes in the corresponding dataset. Next, the dimensionality d^0 in Eq. (5) was set to 6 on each dataset. We used the Adam optimizer and the learning rate in the optimization algorithm was set as 0.011. The maximum number of training epochs was set as 3000. In practice, we made use of PyTorch for an efficient GPU-based implementation of Algorithm 1 using sparse-dense matrix multiplications.[2]

For the baselines, we tuned their settings empirically. First, for GNN-based methods and graph pooling-based methods, the embedding dimension and dropout [67] rate of these models were set to 16, 0.5, respectively. GCN, GAT, MotifNet used their default aggregate function and GraphSAGE used max aggregate empirically. The degree of multivariate polynomial filters in MotifNet was set to 1 and utilizes 13 motif-based adjacency matrices. Considering that GAT concatenates different head outputs, which is similar to MGNN. Therefore, GAT was set to use 13 heads and the embedding dimension is 8. Second, for network embedding-based methods, the embedding dimension of these models was set to 128, and we used the logistic regression model [68] as a classifier to evaluate the quality of the embeddings generated by these unsupervised models. The other settings for these models largely align with the literature.

Note that, in our experiments, all the methods make use of the same directed/undirected edge information on each dataset. Specifically, Chem2Bio2RDF is a directed graph. The baseline implementations used here are able to deal with directed graphs, in which message propagation follows the given edge directions. Meanwhile, ENZYMES, MUTAG, and AIDS are all undirected graphs, and the original Cora, Citeseer and Pubmed are directed citation graphs. Following standard benchmarking practice, the three citation graphs are treated as undirected [69], [70], where a preprocessing step is applied to ignore edge directions for all the methods. Note that when an undirected graph is fed into MGNN, MGNN treats each undirected edge as two directed edges in opposite directions.

We adopt a widely-used *accuracy* metric for performance evaluation. For the node classification task, similar to the experimental setup in [71], we split the dataset into 500 nodes for validation, 500 nodes for testing, and the remaining nodes were used for training, to simulate labeled and unlabeled information. Note that Chem2Bio2RDF is an exception, and we split it into 5000 nodes for validation and 5000 nodes for testing due to its large size. Then we report the average and standard deviation of accuracy scores across the 5 runs with different random seeds. For the graph classification task, similar to the experimental setup in [33], we perform 5-fold cross validation. For other experiments, we present the average accuracy scores over the 5 runs with various random seeds.

B. Performance Evaluation

We evaluate the empirical performance of MGNN against the state-of-the-art baselines in Tables III and IV.

1) Comparison to baselines: As shown in Table III, MGNN significantly and consistently outperforms all the baselines on different datasets. In particular, GraphSAGE achieves the second best performance on Pubmed and Chem2Bio2RDF, while MixHop achieves the second best performance on Cora and Chem2Bio2RDF, and GDC achieves the second best performance on Citeseer. Our MGNN is capable of achieving further improvements against GraphSAGE by 2.81% on Pubmed, against MixHop by 2.92% on Cora, as well as against

[2]Our source codes and pre-processed datasets are publicly available via https://github.com/CXX1113/MGNN.

TABLE III
PERFORMANCE ON THE NODE CLASSIFICATION TASK, MEASURED IN ACCURACY. STANDARD DEVIATION ERRORS ARE GIVEN. THE BEST PERFORMANCE IS MARKED IN BOLD, AND THE SECOND BEST IS UNDERLINED

	Cora		Citeseer		Pubmed		Chem2Bio2RDF	
DeepWalk	0.4313	0.0221	0.2732	0.0216	0.4440	0.0208	0.9253	0.0023
GraRep	0.5957	0.0062	0.4220	0.0022	0.6147	0.0073	0.9313	0.0018
HOPE	0.4510	0.0010	0.3180	0.0021	0.4880	0.0011	0.9030	0.0001
Node2Vec	0.7150	0.0042	0.4670	0.0145	0.6788	0.0063	0.9029	0.0012
GCN	0.8595	0.0207	0.7764	0.0045	0.8865	0.0048	0.9371	0.0017
GraphSAGE	0.8610	0.0101	0.7744	0.0061	0.8980	0.0049	0.9630	0.0010
GAT	0.8775	0.0127	0.7852	0.0052	0.8840	0.0079	0.9628	0.0017
GIN	0.8107	0.0188	0.7255	0.0160	0.8810	0.0156	0.9205	0.0129
BGNN	0.8470	0.0143	0.7750	0.0112	0.8380	0.0119	0.8746	0.0115
MotifNet	0.8580	0.0075	0.7750	0.0071	0.8895	0.0102	0.8863	0.0114
MixHop	0.8803	0.0120	0.7796	0.0053	0.8628	0.0150	0.9630	0.0004
GDC	0.8660	0.0100	0.7854	0.0061	0.8768	0.0059	0.8838	0.0036
CADNet	0.8612	0.0131	0.7652	0.0148	0.8772	0.0085	0.8287	0.0258
MGNN	0.9060	0.0049	0.7948	0.0050	0.9232	0.0084	0.9870	0.0021

GraphSAGE and MixHop by 2.49% on Chem2Bio2RDF. On Citeseer, MGNN outperforms GDC by 1.20% in terms of accuracy. Note that the number of edges in Citeseer is small and the occurrences of motifs are limited. Therefore, our MGNN cannot collect as much high-order information as it can on other datasets, and MGNN achieves less improvement on Citeseer than on other datasets.

Similarly, in Table IV, MGNN regularly surpasses all the baselines. In particular, GCN achieves the second best performance on AIDS, while GDC achieves the second best performance on MUTAG, and MixHop achieves the second best performance on ENZYMES. MGNN is able to achieve further improvements against GCN by 0.76% on AIDS, against GDC by 3.18% on MUTAG and against MixHop by 10.95% on ENZYMES as shown in Table IV. In particular, a graph represents a compound's molecular structure in these three biochemical graph datasets. Any chemical structure can be represented by 13 motifs, which allows our MGNN to identify similar structures among various compounds and boost classification accuracy. For example, both carbon dioxide CO_2 and methane CH_4 have the motif M_8. Moreover, CH_4 has six M_8 while CO_2 has one M_8 only, and such difference is useful for graph classification.

TABLE IV
PERFORMANCE ON THE GRAPH CLASSIFICATION TASK IN TERMS OF ACCURACY. STANDARD DEVIATION ERRORS ARE GIVEN.

	MUTAG		ENZYMES		AIDS	
Graph2Vec	0.6650	0.0087	0.2033	0.0239	0.8045	0.0033
GL2Vec	0.6703	0.0106	0.1967	0.0461	0.8225	0.0065
NetLSD	0.7450	0.0611	0.2136	0.0461	0.9575	0.0082
Feather	0.7716	0.0341	0.2483	0.0226	0.7930	0.0019
Graclus	0.7504	0.0750	0.2567	0.0253	0.8640	0.0398
ASAP	0.7562	0.0799	0.2600	0.0320	0.8960	0.0279
EdgePool	0.7508	0.0687	0.2500	0.0449	0.8615	0.0581
TopKPool	0.7238	0.0527	0.2417	0.0349	0.8530	0.0492
GlobalATT	0.7346	0.0736	0.2383	0.0427	0.8390	0.0248
GCN	0.7555	0.0651	0.2100	0.0285	0.9895	0.0091
GAT	0.7391	0.0315	0.1667	0.0000	0.8740	0.1013
GraphSAGE	0.7984	0.0526	0.2333	0.0586	0.9855	0.0091
GIN	0.7780	0.0940	0.2630	0.0330	0.9870	0.0090
MotifNet	0.8040	0.0330	0.1770	0.0140	0.9880	0.0060
MixHop	0.7663	0.0897	0.2767	0.0494	0.9265	0.0157
GDC	0.8199	0.0849	0.2633	0.0126	0.8705	0.0165
CADNet	0.7450	0.0531	0.2267	0.0273	0.7995	0.0011
MGNN	0.8460	0.0230	0.3070	0.0300	0.9970	0.0030

Next, we further compare the robustness of MGNN and the baseline approaches by introducing noise. Specifically, by replacing the original input node features with a 16-dimensional random vector, we first modified Cora and Pubmed which are denoted as Cora-RandomX and Pubmed-RandomX. Then we compared the performance of MGNN and the baselines on the above two modified datasets. As shown in Fig. 7(a)(b), the performance of MGNN and other GNN baselines on Cora-RandomX and Pubmed-RandomX showed signs of deterioration to varying degrees compared with that on Cora and Pubmed, which is intuitive since additional noise is introduced through random features. Importantly, not only does MGNN considerably and continuously exceed all GNN baselines on the accuracy metric but also its rate of decrease is the lowest. This is because MGNN can grasp more high-order structure information with higher discriminative power, which makes MGNN more robust than other standard or motif-based GNNs.

Fig. 7. (a) and (b) denote the performance of two modified datasets on robustness comparison using random vectors as node features, respectively. Compared to Cora and Pubmed, the performance degradation rate of models is denoted by the inverted black triangle with the percentage on the top of each bar in (a)(b).

Fig. 8. Ablation study on all motifs on seven datasets, minimal-redundancy operator and injective vector concatenation function on three datasets: (a) the ablation study result related to all motifs on seven datasets and (b)(c) the ablation study results related to and concatenation function on three datasets, respectively.

2) Model ablation study: As Fig. 4 illustrates, the network motif, motif redundancy minimization and injective M_k-based representation concatenation are key components in our proposed MGNN. We can thus derive the following variants of MGNN: (1) MGNN without any motif information denoted as MGNNw/oM (this variant is actually a GCN); (2) MGNN without motif redundancy minimization operator denoted as MGNNw/o ; (3) MGNN with other functions for feature vector combination, including summation, max and mean. To show the impact of motifs, and injective concatenation in MGNN, we compare MGNN with the above variants.

In Fig. 8, we observe that MGNN achieves better performance than the variants in terms of accuracy, demonstrating the effectiveness of motifs, and injective concatenation. Firstly, in order to demonstrate the impact of the motif on MGNN's performance, we compare MGNN with the variant MGNNw/oM. As Fig. 8(a) shows, we observe that the performance of MGNNw/oM is lower than that of MGNN on all the 7 datasets, demonstrating the importance of incorporating motif into GNNs, that is, the high-order structures grasped by motif are important for GNNs' performance. Secondly, in order to investigate the impact of, we removed the minimal-redundancy operator on MGNN, and the comparison between MGNN and MGNNw/o is as Fig. 8(b) illustrates. As can be seen, MGNN significantly outperforms MGNNw/o in terms of accuracy in three datasets. Note that the number of edges in Pubmed is large and the redundancy of motifs is probably higher than other datasets (i.e, different motifs in Pubmed share more certain substructures). Therefore, on Pubmed, MGNNw/o is more difficult to distinguish between different motif-wise representations than on other datasets, and the performance gap between MGNN and MGNNw/o is more pronounced on Pubmed than on other datasets. Thirdly, to demonstrate the impact of injective concatenation, we used other non-injective vector combination functions, including summation, max, and mean, to replace injective concatenation. Fig. 8(c) illustrates that MGNN with concatenation performs significantly better than MGNN with other functions on these three datasets. Moreover, we show the results of different combination functions (namely, concatenation, max, sum and mean) on three datasets in Table V. We can observe that the performance decline is larger on Citeseer and Cora than on PubMed. A potential reason is Cora and Citeseer are very sparse and the occurrences of motifs are limited. Thus, the limited number of motifs on Cora and Citeseer would make it more difficult to distinguish among different node representations using non-injective functions.

TABLE V
PERFORMANCE OF MGNN BY USING DIFFERENT COMBINATION FUNCTIONS ON THREE DATASETS. THE RATES OF DECLINE IN PERFORMANCE W.R.T. CONCATENATION ARE GIVEN IN PARENTHESES

	Concat	Max	Sum	Mean
Cora	0.906	0.878 (3.09%)	0.884 (2.43%)	0.894 (1.32%)
Citeseer	0.795	0.776 (2.37%)	0.766 (3.62%)	0.788 (0.86%)
Pubmed	0.923	0.914 (1.00%)	0.918 (0.56%)	0.918 (0.56%)

C. Case Study

In this section, we investigate the importance of different motifs for prediction and demonstrate the necessity of using the high-order structure for prediction. We completed the following two studies on the Chem2Bio2RDF dataset. First, MGNN makes a prediction across the 5 runs by using only 1 out of the 13 motifs and then compares the results to determine the significance of various motifs. Second, we take protein-disease association prediction as our case study. In particular, we rank the protein-disease pairs based on their predicted scores, then identify those top pairs supported by existing publications. Meanwhile, we also evaluate the performance of MGNN versus the baseline approaches for protein-disease association prediction. Next, we show the details of these two studies.

1) The importance of different motifs for prediction: To demonstrate the importance of different motifs, MGNN utilizes just one motif to conduct node classification across the 5 runs on the Chem2Bio2RDF network, and the performance of MGNN is used to assess the importance of each motif in Table VI.

As shown in Table VI, we can draw two conclusions. First, for a Chem2Bio2RDF network, the importance of different motifs varies. This is because important motifs often serve as building blocks within a network, and can even be used to define universal classes of the network they are in [12]. For example, M_{13} is a building block of the protein-protein interaction (PPI) network on the Chem2Bio2RDF graph (protein-protein-protein), and the PPI network is key for protein-disease association prediction. M_{13} is thus ranked as one of top 3 most significant motifs on this dataset as shown in Table VI. Another example is the triangular motifs (M_1-M_7), which are essential in social networks due to their

TABLE VI
IMPORTANCE RANKING OF 13 MOTIFS ON CHEM2BIO2RDF. THE IMPORTANCE SCORE OF EACH MOTIF IS THE PERFORMANCE OF MGNN WHEN USING ONLY THAT MOTIF FOR PREDICTION. THE SYMBOL `ALL' INDICATES THAT ALL MOTIFS ARE USED BY MGNN.

Rank	Motif	ACC	Rank	Motif	ACC
1	M3	0.9809	8	M1	0.9686
2	M13	0.9802	9	M8	0.9477
3	M7	0.9791	10	M10	0.9408
4	M2	0.9789	11	M12	0.9377
5	M11	0.9781	12	M9	0.9317
6	M5	0.9762	13	M6	0.9271
7	M4	0.9717	-	ALL	0.9870

triadic closure nature [13], [72]. Second, the performance of the top three motifs is similar to the performance of all motifs combined (last row on the right). This is because a single motif may effectively encapsulate all of the network's essential information. With these two conclusions, we can see that one of the advantages of MGNN is its generality. That is, even if the importance of motifs is unknown, we can still use MGNN with all the motifs to achieve a final performance similar to that of using important motifs only.

To further demonstrate the adaptive selection results of our motif redundancy minimization operator, we conduct the following experiments on Chem2Bio2RDF. We randomly sample 15 nodes and presented the representations of the top 3 and bottom 3 most significant motifs in Table VI by heatmap. As shown in Fig. 9, representations w.r.t. unimportant motifs (M_9, M_6) are more sparse than other motifs. In addition, we observed that there are often only no more than three non-zero dimensions for a motif, which shows that MGNN actually needs a very low dimension to capture high-order structures.

2) The necessity of incorporating high-order structure information for prediction: We take protein-disease association prediction on the Chem2Bio2RDF dataset as an example to demonstrate the necessity of incorporating higher-order information from two aspects, i.e., illustration of validity and illustration of practicality. For an illustration of validity, we train MGNN and compare it to the baseline methods for protein-disease association prediction in terms of the area under the ROC curve (AUROC). For an illustration of practicality, we rank all the protein-disease pairs based on their predicted scores and then identify top pairs supported by existing publications.

Fig. 9. Heatmap of the top 3 (M_3; M_{13}; M_7) and bottom 3 (M_{12}; M_9; M_6) most important motif-based representations on Chem2Bio2RDF.

Fig. 10. Performance on protein-disease association prediction in Chem2Bio2RDF dataset, measured in AUROC. Standard deviation errors are given.

TABLE VII
TOP PREDICTED PROTEIN–DISEASE ASSOCIATIONS WITH LITERATURE SUPPORT

Rank	Gene	Disease	PubMed ID
1	COX2	Colorectal Carcinoma	26 159 723
6	CTNNB1	Colorectal Carcinoma	24 947 187
7	P2RX7	Colorectal Carcinoma	28 412 208
12	SMAD3	Colorectal Carcinoma	30 510 241
16	HRH1	Colorectal Carcinoma	30 462 522
37	ABCB1	Colorectal Carcinoma	28 302 530
44	AKT1	Malignant neoplasm of breast	29 482 551
64	TP53	Malignant neoplasm of breast	31 391 192
82	EP300	Colorectal Carcinoma	23 759 652
92	ADORA1	Colorectal Carcinoma	27 814 614
107	REN	Renal Tubular Dysgenesis	21 903 317
141	FGFR2	Autosomal Dominant	16 141 466
157	BCL2	Non-Hodgkin Lymphoma	29 666 304
169	NOS2	Malignant neoplasm of breast	20 978 357
263	CTNNB1	Mental retardation	24 614 104

MGNN is first trained to predict each protein-disease pair's associated score and compare it to the baseline approaches. Specifically, we will describe this step in detail by stating the background of the task as well as the specific experimental setup. As for the background of the task, protein-disease association prediction is a significant issue with the potential to give clinically actionable insights for disease diagnosis, prognosis, and treatment [73]. The issue can be defined as predicting which proteins are associated with a given disease. Experimental methods and computational methods are the two primary kinds of current attempts to solve this challenge. Experimental methods for gene–disease association, such as genome-wide association studies (GWAS), and RNA interference (RNAi) screens, are costly and time-consuming to conduct. Therefore, a variety of computational methods have been developed to discover or predict gene–disease associations, including text mining, network-based methods [74], and so on. Among them, network-based methods often need to use the structure information of the PPI network (constructed by M_{13} motif). However, high-order PPI network structure is largely ignored in protein-disease discovery nowadays [73]. Our MGNN thus can overcome this limitation.

We next describe the experimental setup. We mapped a protein to the gene that it is produced by, and viewed protein-disease association prediction as a link prediction task on

the graph [73]. We split the edges of the Chem2Bio2RDF dataset with the ratio of 85%/5%/10% for training, validation and testing respectively. We adopted an inner product decoder for link prediction. The parameters of the model were optimized using negative sampling and cross-entropy loss and we used AUROC as a metric. The number of the epoch was set to 1000 and the other hyperparameter settings are consistent with the node/graph classification task. Note that, the Chem2Bio2RDF dataset is missing a semantic mapping to disease IDs. To alleviate this problem, we search for genes associated with the disease (2929 known gene(protein)-disease links in Chem2Bio2RDF), and perform gene-disease association queries in the public database DisGeNET[3] so as to realize the inference of the actual semantics of the disease IDs. Fig. 10 compares the performance of MGNN and other GNN-based methods under five random seeds. As can be seen, MGNN exceeds all GNN baselines on the AUROC metric. Importantly, the AUROC of MGNN is close to 100%, and the standard deviation is very small, which means that MGNN has strong practicability in protein-disease association prediction.

For an illustration of practicality, we further ranked the whole unknown protein-disease pairs (over 28 million unknown pairs) based on their predicted scores, and identified 103 out of the top 1000 pairs that are supported by existing publications. Table VII displays the first 15 of these 103 pairs, and the last column provides the PubMed ID of the publications that support our prediction.

As shown in Table VII, all pairs have been validated by wet-labs and can be found in the DisGeNET database, e.g., row 2 (CTNNB1, Colorectal Carcinoma) is validated by RNAi screening [75], and row 4 (SMAD3,Colorectal Carcinoma) is validated by GWAS [76]. These methods predict protein-disease association from the angles which are orthogonal from MGNN. Therefore, we consider that they provide reasonable supports for our prediction. Taking the first protein-disease pair as an example, COX2 and Colorectal Carcinoma are reported in [77] (i.e. PubMed ID: 26159723). In fact, COX2 is preferentially expressed in cancer cells and its expression is enhanced by proinflammatory cytokines and carcinogens [77]. It is thus reasonable to predict a protein-disease association between them because there is evidence that the over-expression of COX2 is related to the infiltrating growth of Colorectal Carcinoma and other pathological characteristics [78].

D. Parameter Sensitivity

We present the sensitivity analysis for the dimensionality parameters d_i^0 and d_1 in our MGNN.

In Fig. 11(a), the performance of MGNN is not sensitive to changes in the dimensionality d_i^0 in Eqs. (5)–(6). Particularly, values of d_i^0 in the range $[2^2; 2^5]$ typically give a robust and reasonably good performance, e.g., $d_i^0 = 6$ is a desirable choice in most cases. For the output dimensionality d_1 in Eq. (3), as shown in Fig. 11(b), the performance gradually improves and becomes stable around 2^4, which is the preferred choice in most cases.

[3]https://www.disgenet.org/

Fig. 11. Parameter sensitivity analysis for MGNN: (a) dimensionality d_i^0 in Eqs. (5)–(6). (b) output dimensionality d_1 in Eq. (3).

E. Model Size and Efficiency

TABLE VIII
MODEL SIZE AND EFFICIENCY ANALYSIS ON THE PUBMED DATASET.

	# Params	Training per epoch/s	Training overall/min	Inference /ms	Accuracy
GAT	104,624	0.01	1.13	4.00	0.8840
MixHop	24,144	0.02	15.01	19.48	0.8628
BGNN	9,866,596	1.97	608.64	4.57	0.8380
EGAT	108,107	3.72	320.29	20.63	0.8970
ESAGE	212,693	3.11	14.01	5.77	0.9040
EGAT+SAGE	164,443	3.27	150.07	12.34	0.8992
MGNN	26,084	0.04	10.00	1.25	0.9232

We evaluate the model size and efficiency of MGNN, in terms of the number of trainable parameters, training time (per epoch and overall), and inference time. We select a representative baseline from standard GNNs (i.e., GAT) and high-order GNNs (i.e., MixHop), respectively, for comparison to MGNN. Moreover, since MGNN can be viewed as a model that integrates several motif-based modules, we also compare an ensemble GNN here (i.e., BGNN).

For a more comprehensive comparison, we also develop a simple ensemble framework over 13 GNN modules (corresponding to our 13 motifs). First, we separately apply thirteen GNN modules that employ different initializations but otherwise the same input, and fuse their output by a fully connected layer. All hidden dimensions are set to 16. For the above framework, we develop three variants, respectively, the modules use only 13 GATs, only 13 GraphSAGEs as well as 7 GraphSAGEs and 6 GATs, denoted as EGAT and ESAGE, and EGAT+SAGE, respectively.

As shown in Table VIII, MGNN is competitive in terms of model size and efficiency, while achieving the best accuracy. In particular, although several ensemble methods including EGAT, ESAGE and EGAT+SAGE achieve better accuracies among the baselines, their model sizes or efficiency are all worse than MGNN. Note that the per epoch and overall training times are often inconsistent across methods, as a method may train faster per epoch but it converges slower, or vice versa. Further experiments involving ensemble GNNs on all datasets are presented in Section II of our supplementary materials.

VII. Conclusion

We propose Motif Graph Neural Networks, a novel framework to better capture high-order structures. Different from previous work, we propose the motif redundancy minimization operator and injective motif combination to improve the discriminative power of GNNs on the high-order structure. We also propose an efficient manner to construct a motif-based adjacency matrix. Further, we theoretically show that MGNN is provably more expressive than standard GNN, and standard GNN is in fact a special case of MGNN. Finally, we demonstrate that MGNN outperforms all baselines on seven public benchmarks.

References

[1] W. Fan, Y. Ma, Q. Li, Y. He, J. T. Yihong Eric Zhao, and D. Yin, "Graph neural networks for social recommendation," in Proc. WWW, 2019, pp. 417–426.

[2] M. Sun, S. Zhao, C. Gilvary, O. Elemento, J. Zhou, and F. Wang, "Graph convolutional networks for computational drug development and discovery," Briefings in bioinformatics, pp. 919–935, 2020.

[3] B. Xu, H. Shen, B. Sun, R. An, Q. Cao, and X. Cheng, "Towards consumer loan fraud detection: Graph neural networks with role-constrained conditional random field," in Proc. Association for the Advancement of Artif. Intell., 2021, pp. 4537–4545.

[4] J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics," Machine Learning: Science and Technology, 2020.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in International Conference on Learning Representations, 2017.

[6] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," CoRR, 2017.

[7] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," Advances in Neural Inf. Processing Systems, pp. 1024–1034, 2017.

[8] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 1105–1114.

[9] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in CIKM, 2015, pp. 891–900.

[10] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in SIGKDD, 2016, pp. 855–864.

[11] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in SIGKDD 2014, S. A. Macskassy, C. Perlich, J. Leskovec, W. Wang, and R. Ghani, Eds. ACM, 2014, pp. 701–710.

[12] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: simple building blocks of complex networks," Science, 2002.

[13] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," Science, 2016.

[14] J. B. Lee, R. A. Rossi, X. Kong, S. Kim, E. Koh, and A. Rao, "Graph convolutional networks with motif-based attention," in CIKM 2019, W. Zhu, D. Tao, X. Cheng, P. Cui, E. A. Rundensteiner, D. Carmel, Q. He, and J. X. Yu, Eds. ACM, 2019, pp. 499–508.

[15] H. Zhao, Y. Zhou, Y. Song, and D. L. Lee, "Motif enhanced recommendation over heterogeneous information network," in CIKM 2019, W. Zhu, D. Tao, X. Cheng, P. Cui, E. A. Rundensteiner, D. Carmel, Q. He, and J. X. Yu, Eds. ACM, 2019, pp. 2189–2192.

[16] F. Monti, K. Otness, and M. M. Bronstein, "MOTIFNET: A motif-based graph convolutional network for directed graphs," in 2018 IEEE Data Science Workshop, DSW 2018. IEEE, 2018, pp. 225–228.

[17] A. Sankar, J. Wang, A. Krishnan, and H. Sundaram, "Beyond localized graph neural networks: An attributed motif regularization framework," in ICDM 2020, 2020, C. Plant, H. Wang, A. Cuzzocrea, C. Zaniolo, and X. Wu, Eds. IEEE, 2020, pp. 472–481.

[18] H. Zhao, X. Xu, Y. Song, D. L. Lee, Z. Chen, and H. Gao, "Ranking users in social networks with higher-order structures," in Proc. Association for the Advancement of Artif. Intell., 2018, pp. 232–240.

[19] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in Int. Conf. on Machine Learning, 2017, pp. 1263–1272.

[20] P. W. Battaglia, J. B. Hamrick, V. Bapst, and A. Sanchez-Gonzalez, "Relational inductive biases, deep learning, and graph networks," 2018.

[21] J. Gan, R. Hu, Y. Mo, Z. Kang, L. Peng, Y. Zhu, and X. Zhu, "Multigraph fusion for dynamic graph convolutional network," IEEE Transactions on Neural Networks and Learning Systems, 2022.

[22] Y. He, D. Yan, W. Xie, Y. Zhang, Q. He, and Y. Yang, "Optimizing graph neural network with multiaspect hilbert-schmidt independence criterion," IEEE Transactions on Neural Networks and Learning Systems, 2022.

[23] M. Gong, H. Zhou, A. Qin, W. Liu, and Z. Zhao, "Self-paced co-training of graph neural networks for semi-supervised node classification," IEEE Transactions on Neural Networks and Learning Systems, 2022.

[24] D. Guo, C. Xu, and D. Tao, "Bilinear graph networks for visual question answering," neural networks and learning systems, 2021.

[25] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," in ICLR, 2016.

[26] Y. Li, D. Tarlow, M. Brockschmidt, and R. S. Zemel, "Gated graph sequence neural networks," in ICLR, 2016.

[27] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in Proc. Association for the Advancement of Artificial Intelligence, 2018, pp. 4438–4445.

[28] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors A multilevel approach," IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, pp. 1944–1957, 2007.

[29] P. P. T. Ekagra Ranjan, Soumya Sanyal, "ASAP: adaptive structure aware pooling for learning hierarchical graph representations," in Proc. Association for the Advancement of Artificial Intelligence, 2020, pp. 5470–5477.

[30] F. Diehl, T. Brunner, M. T. Le, and A. Knoll, "Towards graph pooling by edge contraction," in International Conference on Machine Learning, 2019.

[31] H. Gao and S. Ji, "Graph u-nets," in International Conference on Machine Learning, 2019, pp. 2083–2092.

[32] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in Proc. Neural Information Processing Systems, 2018, pp. 4805–4815.

[33] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in ICLR, 2019.

[34] K. Zhang, "Capturing high-order structures on motif-based graph nerual networks," CoRR, 2022.

[35] M. R. Dareddy, M. Das, and H. Yang, "motif2vec: Motif aware node representation learning for heterogeneous networks," in Big Data, 2019.

[36] A. Sankar, X. Zhang, and K. C. Chang, "Motif-based convolutional neural network on graphs," CoRR, vol. abs/1711.05697, 2017.

[37] B. Wang, L. Cheng, J. Sheng, and Z. Hou, "Graph convolutional networks fusing motif-structure information," Scientific Reports, 2022.

[38] Y. Yang, Z. Guan, W. Zhao, L. Weigang, and B. Zong, "Graph substructure assembling network with soft sequence and context attention," IEEE Transactions on Knowledge and Data Engineering, 2022.

[39] H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman, "Provably powerful graph networks," in Proc. Neural Information Processing Systems, 2019, pp. 2153–2164.

[40] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in Proc. Association for the Advancement of Artificial Intelligence, 2019, pp. 4602–4609.

[41] C. Morris, G. Rattan, and P. Mutzel, "Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings," in Proc. Neural Information Processing Systems, pp. 21 824–21 840, 2020.

[42] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in International Conference on Machine Learning, 2019, pp. 21–29.

[43] J. Klicpera, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," in Proc. Neural Information Processing Systems, 2019, pp. 13 333–13 345.

[44] J. Lim, D. Um, H. J. Chang, D. U. Jo, and J. Y. Choi, "Class-attentive diffusion network for semi-supervised classification," in Proc. Association for the Advancement of Artificial Intelligence, 2021, pp. 8601–8609.

[45] D. Flam-Shepherd, T. C. Wu, P. Friederich, and A. Aspuru-Guzik, "Neural message passing on high order paths," Machine Learning: Science and Technology, vol. 2, 2021.

[46] T. Zhang, Q. Wu, and J. Yan, "Learning high-order graph convolutional networks via adaptive layerwise aggregation combination," Neural Networks and Learning Systems, 2021.