12-2024

Harnessing collective structure knowledge in data augmentation for graph neural networks

Rongrong MA

Guansong PANG
Singapore Management University, gspang@smu.edu.sg

Ling CHEN

Citation

MA, Rongrong; PANG, Guansong; and CHEN, Ling. Harnessing collective structure knowledge in data augmentation for graph neural networks. (2024). *Neural Networks*. 180, 1-13.
Available at: https://ink.library.smu.edu.sg/sis_research/9289

Harnessing Collective Structure Knowledge in Data Augmentation for Graph Neural Networks

Rongrong Ma[a], Guansong Pang[b,*], Ling Chen[c]

[a] *Faculty of Engineering and Information Technology, University of Technology Sydney, 123 Broadway, Sydney, 2007, NSW, Australia*
[b] *School of Computing and Information Systems, Singapore Management University, 80 Stamford Rd, 178902, Singapore*
[c] *Faculty of Engineering and Information Technology, University of Technology Sydney, 123 Broadway, Sydney, 2007, NSW, Australia*

Abstract

Graph neural networks (GNNs) have achieved state-of-the-art performance in graph representation learning. Message passing neural networks, which learn representations through recursively aggregating information from each node and its neighbors, are among the most commonly-used GNNs. However, a wealth of structural information of individual nodes and full graphs is often ignored in such process, which restricts the expressive power of GNNs. Various graph data augmentation methods that enable the message passing with richer structure knowledge have been introduced as one main way to tackle this issue, but they are often focused on individual structure features and difficult to scale up with more structure features. In this work we propose a novel approach, namely collective structure knowledge-augmented graph neural network (CoS-GNN), in which a new message passing method is introduced to allow GNNs to harness a diverse set of node- and graph-level structure features, together with original node features/attributes, in augmented graphs. In doing so, our approach largely improves the structural knowledge modeling of GNNs in both node and graph levels, resulting in substantially improved graph representations. This is justified by extensive empirical results where CoS-GNN outperforms state-

[*] Corresponding author

Email addresses: Rongrong.ma-1@student.uts.edu.au (Rongrong Ma), gspang@smu.edu.sg (Guansong Pang), ling.chen@uts.edu.au (Ling Chen)

Preprint submitted to Neural Networks *May 20, 2024*

of-the-art models in various graph-level learning tasks, including graph classification, anomaly detection, and out-of-distribution generalization. Code is available at: `https://github.com/RongrongMa/CoS-GNN`.

Keywords: Graph representation learning, Graph neural networks, Data augmentation

1. Introduction

Graph representation learning is one of the most popular topic in graph mining because of its numerous potential applications in bioinformatics [1, 2, 3, 4, 5], chemical [6, 7, 8], social networks [9, 10, 11, 12] and cyber security [13]. In the past few years, Graph Neural Networks (GNNs) [14, 43] have been emerging as one of the most powerful and successful techniques for graph representation learning.

Message passing neural networks constitute a prevalent category of GNN models, which learn node features and graph structure information through recursively aggregating current representations of node and its neighbors. Diverse aggregation strategies have been introduced, giving rise to various GNN backbones, such as GCN, GIN, and among others [14, 15, 16, 17, 18]. However, the expressive power of these message passing GNNs is upper bounded by 1-dimensional Weisfeiler-Leman (1-WL) tests [18, 19] that encode a node's color via recursively expanding the neighbors of the node to construct a rooted subtree for the node. As shown in Figure 1, such rooted subtrees are with limited expressiveness and might be the same for graphs with different structures, leading to failure in distinguishing these graphs. This presents a bottleneck for applying WL tests or message passing neural networks to many real-world graph application domains.

The failure of WL test is mainly due to the rooted subtree's limited capabilities in capturing different substructures that can appear in the graph. Since the message passing scheme of GNNs mimics the 1-WL algorithm, one intuition to enhance the expressive power of GNNs is to enrich the passing information, es-

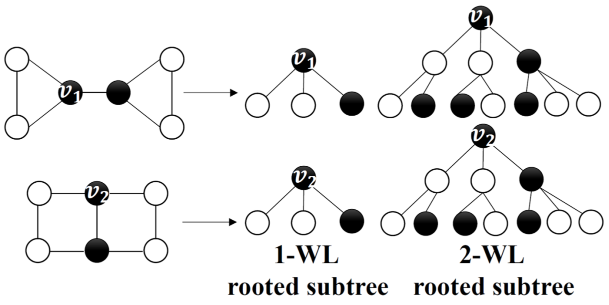
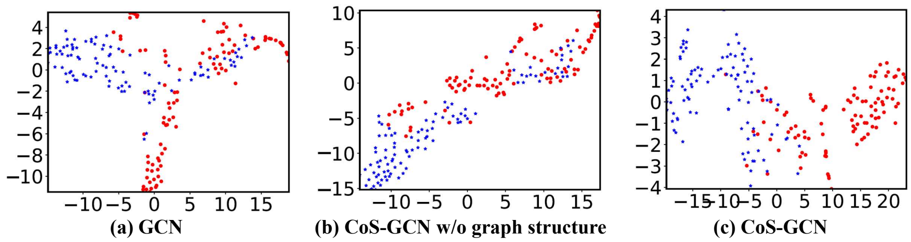


Figure 1: 1- and 2-WL tests fail to distinguish the two graphs as they obtain the same rooted subtree (node coloring).

pecially structural knowledge, to help GNNs model diverse substructures. One popular approach to achieve this is data augmentation (DA) techniques [20]. One general framework in this line is to compute additional node features based on structural properties and attach them to original node features, such as DE [21], GSN [22], fast ID-GNN [23] and LAGNN [24]. Except extending node features, NestedGNN [25] and ID-GNN [23] compute and add node embeddings based on the local subgraph of each node. However, these methods only focus on local structure while many important global structure features are ignored. Also, GSN and fast ID-GNN often rely on a properly pre-defined substructure set to incorporate domain-specific inductive biases [25]. Further, these DA techniques are focused on augmenting the graph with some individual features, which are difficult to scale up to the incorporation of a diverse, large set of augmented features.

In this work, we propose a novel approach, namely *collective structure knowledge-augmented graph neural network (CoS-GNN)*, to leverage a variety of informative structural knowledge of graphs through DA for enhancing the expressiveness of existing GNNs. Instead of implicitly using structural information in other DA methods, we explicitly extract collective, domain-adaptive graph structural statistics at the graph and node levels as additional structure features. To fully leverage those augmented structural knowledge, we design a new message passing mechanism to respectively perform neighborhood aggregation on graph



(a) GCN (b) CoS-GCN w/o graph structure (c) CoS-GCN

Figure 2: Graph representations of REDDIT-BINARY yielded by (b) CoS-GCN with augmented node-level structural features and (c) CoS-GCN with augmented structural features at both node and graph levels are more class-separable than those produced by (a) the original GCN.

data using these augmented structure features and the original node attributes. Further, the new message passing can also model the interaction between the augmented features and the original node attributes. In doing so, our GNNs break down the upper bound of 1-WL tests and learn graph representations with significantly improved expressiveness (see the graph representations produced by CoS-GNN in Figure 2(b)(c) vs. those yielded by the original GCN).

In summary, our main contributions are as follows:

- We introduce a novel collective structure knowledge augmented GNN approach (CoS-GNN) that explicitly harnesses a diverse set of node and graph-level structural information for enhancing the expressiveness of GNN-based graph representations. The approach is generic and applicable to different GNN backbones.

- To effectively leverage the augmented structural features, a new message passing scheme is introduced in CoS-GNN, which simultaneously performs neighborhood aggregation on the augmented features and the original node attributes, enabling the learning of graph representations with significantly enriched structural knowledge.

- Comprehensive experiments on 12 graph datasets demonstrate that CoS-GNN (i) significantly outperforms competing methods in graph classification task; (ii) provides more discriminative information for anomaly detection task; (iii) is more generalized to out-of-distribution graphs.

2. Related Work

2.1. Graph Neural Networks

Graph neural networks (GNNs) have gained numerous attentions and remarkable success in the past few years [14, 37, 38, 43]. Wu et al. [14] summarize various GNN models, among which message passing between nodes based on graph structure to learn graph representations is one of the most popular ones. They iteratively aggregate information from nodes and their neighbors to learn expressive representations of nodes. GCN [15], GraphSAGE [16], graph attention network (GAT) [17] and graph isomorphism network (GIN) [18] are several most representative and state-of-the-art message-passing GNNs. However, since the message passing mechanism of GNNs mimics the 1-WL algorithm, the expressive power of these GNNs is limited and upper bounded by the Weisfeiler-Leman test [18, 19], which restricts their performance in graph representation learning. The proposed CoS-GNN aims to improve the expressiveness of these popular GNNs by modeling a collective set of additional structural features.

2.2. Graph Data Augmentation

Motivated by the excellent achievements of data augmentation in image and text data [26, 27], graph data augmentation techniques are attracting increasing attention to improve the representation expressiveness obtained from GNNs in graph-domain tasks [20]. To solve the label scarcity in semi/unsupervised classification tasks, augmentation through changing the graph structure or node attribute, *e.g.*, node/edge deletion, attribute masking, and subgraph sampling, is used to construct a contrastive or consistent learning framework in [39, 40, 41, 42, 44]. In supervised graph classification tasks, some augmentation methods are implemented to enhance the representation expressiveness of GNNs and further improve the classification performance. One direction is to enrich node features. For example, distance encoding is proposed in

[21] to generate and add extra node features. Sato, Yamada and Kashima [28] add random node features, while GSN [22] and fast ID-GNN [23] use the count of various motifs to extend the node features. In [24], neighborhood features are augmented via a generative model conditioned on local structures and node features. Except to node feature extension, augmenting the graph from the structure perspective is another popular approach. For example, Zhang and Li [25] and You et al. [23] sample a subgraph for each node and use the subgraphs to compute node embeddings and add them to complement the original graph. The structural information used in these methods is local, while many useful graph-level structural information is ignored. To alleviate this issue, Liu et al. [29] add a dummy node that connects to all existing nodes without affecting original node and edge properties for better graph representation learning. $\mathcal{G}$-mixup [30] and graph transplant [31] mixup graphs to obtain more graph data for data-hungry tasks. Papp et al. [32] instead perform augmentation through iteratively removing nodes randomly and executing multiple different runs on these node-dropout graphs. All these DA methods are focused on utilizing some specific types of additional features, like substructure and local structure variation, which often cannot generalize to graphs from different application domains. By contrast, our approach aims to harness collective, domain-adaptive node and graph features via a new message passing mechanism.

3. Framework

3.1. Problem Statement

This work focuses on the problem of graph representation learning. Specifically, given a set of graphs $\mathcal{G} = \{G_1, \cdots, G_N\}$, each graph $G = \{\mathcal{V}_G, \mathcal{E}_G\}$ contains a vertex/node set $\mathcal{V}_G$ and an edge set $\mathcal{E}_G$. The structure of G is denoted by an adjacency matrix $A \in \mathrm{R}^{|\mathcal{V}_G| \times |\mathcal{V}_G|}$, where $|\mathcal{V}_G|$ is the number of nodes in G. $A(i,j) = 1$ if an edge exists between node v_i and v_j ($\exists (v_i, v_j) \in \mathcal{E}_G$), and $A(i,j) = 0$ otherwise. If a feature vector $\mathbf{x}_i^n \in \mathrm{R}^d$ is associated with each node $v_i \in \mathcal{V}_G$, the graph G is an attributed graph, and otherwise G is a plain graph.

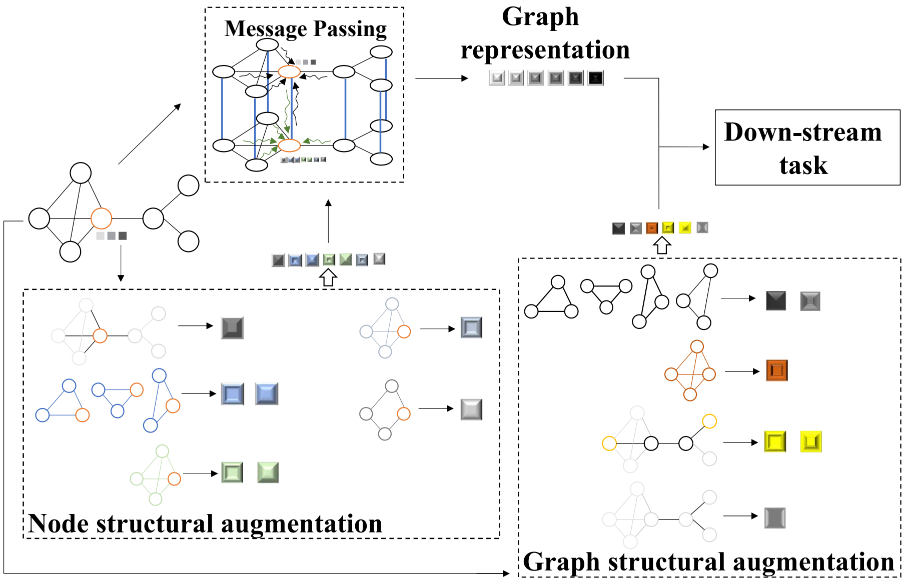


Figure 3: A schematic depiction of our CoS-GNN. Our CoS-GNN first calculates the specific node- and graph-level structural features. Then a new message passing mechanism is devised to utilize the original node attributes and the augmented node structural features to compute the graph representation, which is further combined with the graph structural augmentations for down-stream tasks.

For a plain graph, we use the one-hot node label as the node attributes. Our goal is to learn a representation for each graph G for further use in down-stream tasks like graph classification and anomaly detection.

3.2. The Proposed Framework

We propose a novel collective structure knowledge-augmented graph neural network (CoS-GNN) that aggregates original and augmented structural features of single nodes and whole graph to learn expressive graph representations. The key intuition of CoS-GNN is to utilize various local (node) and global (graph) structural information to enrich the original graph structural knowledge, through which we can learn a more informative and discriminative graph representation. The overall procedure of CoS-GNN is illustrated in Figure 3, which is composed of the following three major components:

- *Collective Graph Data Augmentation.* In this component, we generate a diverse set of specific structural features for each graph G (denoted by $\mathbf{x}_G^{gs}$) and each node v_i in G (denoted by $\mathbf{x}_i^{ns}$). These two types of features are added to augment each graph G from the structural knowledge perspective. It is a component that can be done offline.

- *Augmented Node-level Message Passing.* This component is designed to iteratively aggregate both the original and augmented node features, *i.e.*, $\mathbf{x}_i^n$ and $\mathbf{x}_i^{ns}$, to learn the node representation $\mathbf{h}_i$ with significantly enriched structural knowledge for each node v_i. To this end, a new message passing mechanism is introduced for this process. The node representations are then fed to a readout layer to gain the graph representation $\mathbf{h}_l$.

- *Graph-level Representation Fusion.* This component aims to synthesize the learned graph representation $\mathbf{h}_l$ and the pre-defined graph-level structural features $\mathbf{x}_G^{gs}$ via concatenation/fully-connected layers to obtain the final representation $\mathbf{h}_g$. $\mathbf{h}_g$ is then fed to a down-stream graph-level learning task.

4. Model

CoS-GNN is a generic framework. In this section, we introduce two instantiations of our CoS-GNN framework with the commonly-used GCN and GIN as the GNN backbone, namely CoS-GCN and CoS-GIN, respctively.

4.1. Collective Graph Data Augmentation

We first augment the graph via computing some important node and graph statistics, which serve as additional node and graph features to complement the original node attributes. This component is shared by different model instantiations, and it can be performed before the model training.

Specifically, we select and generate a number of widely-used and domain-adaptive node-level features, including the degree, triangle number, clique size, clique number, core number, cluster coefficient and square cluster coefficient, resulting seven new features in $\mathbf{x}_i^{ns}$ for each node v_i. The last two coefficient measures capture the tendency of the node to form relatively dense communities, while other measures are to capture substructural information from varying scales. The detailed definition of these features is as follows:

- **Degree.** The degree of a node/vertex is the number of edges that are incident to the node, which is an important and commonly-used node structure statistic.
- **Triangle.** Triangle is a simple and direct structure, and we counts the number of triangles that use this node as a vertex.
- **Clique.** The clique is a substructure, in which every two distinct nodes are adjacent. We calculate the size of the maximal clique and the number of maximal cliques containing each given node.
- **K-core.** A k-core is defined as a maximal subgraph that is composed of nodes with degree k or more, the core number of a node is the largest value k of a k-core containing the given node. We collect the core number of each node as one of the augmented node-structural characteristics.
- **Quantized values.** Beyond the number, we also calculate the triangle/square clustering coefficient for each node, which are the fraction of possible triangles/squares through the given node that exist. This quantifies the tendency of nodes to form relatively dense network groups, *i.e.*, triangles or squares.

For graph-structural-level augmentation, we utilize a variety of important global statistics, including triangle number, clique size, the existence of bridge, average clustering coefficient, average global efficiency, and average local efficiency, to generate six graph-level structural features $\mathbf{x}_G^{gs}$ for each graph G. The three coefficients quantify the abundance of dense communities in the graph and the other statistics are the measurement of the node-to-node communication effectiveness within a graph. Detailed definition of each statistic is presented as follows:

- **Triangle.** We use the total number of triangles as one graph feature.
- **Clique.** We count the size of the largest clique in the graph as the second graph feature.
- **Bridge.** Another employed statistic is the existence of a bridge in the graph, which is an edge whose removal will cause the number of connected components of the graph to increase. The bridge is a specific characteristic of the graph.
- **Quantized values.** The average clustering coefficient for the graph is also included to measure the abundance of dense network groups in the graph. The efficiency of a pair of nodes is the multiplicative inverse of the shortest path distance between the nodes, and we calculate the average efficiency of all pairs of nodes in the graph, called average global efficiency, as one of the graph-structural statistics to measure the effectiveness of communication in the graph. The local efficiency of a node is defined as the average global efficiency of the subgraph induced by the neighbors of the node. We utilize the average local efficiency, which is the mean of local efficiencies of each node in the graph, as another statistic.

These collective statistics consider the configurations with different scales and complexities, which are normally adaptive to graphs from different domains.

4.2. Augmented Node-level Message Passing

Once the augmented node structural feature $\mathbf{x}^{ns}$ is obtained, we then aggregate the original feature $\mathbf{x}^{n}$ and the augmented features $\mathbf{x}^{ns}$ to learn the original node attributes and their interaction with augmented structural knowledge of nodes. One straightforward solution that many previous methods do is to concatenate them directly and then apply GNN to perform the commonly-used neighborhood aggregation on nodes using the combined feature. This approach is easy-to-implement but fails to capture intricate interactions (e.g., higher-order and/or non-linear interactions) between the original node attributes and augmented features. To address this issue, we propose a novel message passing mechanism for effectively capturing the diverse knowledge embedded in the two types of features and their interactions. Our experiments also show that our message passing mechanism outperforms the conventional message passing with the concatenated input (see results in Table 10).

To this end, we construct a dual-graph structure that facilitates the modeling of the original node features, the modeling of the collective augmented node features, and the modeling of the interactions between these two types of features in each message passing step. In detail, given a graph G, we construct a new graph $\hat{G}$ with the same node and structure as the original graph but with the $\mathbf{x}^{ns}$ as its node attributes and link the corresponding nodes of G and $\hat{G}$. This results in our augmented graph with a dual-graph structure, $G^{'}$.

4.2.1. Message Passing in CoS-GCN

Next we perform message passing on the dual-graph structure $G^{'}$. When using GCN as our GNN backbone, the adjacent matrix $A^{'}$ of $G^{'}$ can be written as

$$A^{'} = \begin{pmatrix} A & I \\ I & A \end{pmatrix}, \tag{1}$$

and the degree matrix $D^{'}$ is

$$D^{'} = \begin{pmatrix} D+I & 0 \\ 0 & D+I \end{pmatrix}, \tag{2}$$

where A and D are the adjacent and degree matrices of G and I is the identity matrix. We then convolute the node features of $G^{'}$ by

$$\begin{aligned}
H^{(l)} &= \sigma\left({\tilde{D}^{'}}^{-\frac{1}{2}} \tilde{A}^{'} {\tilde{D}^{'}}^{-\frac{1}{2}} \begin{pmatrix} H_n^{(l-1)} \\ H_{ns}^{(l-1)} \end{pmatrix} W^{(l)} \right) \\
&= \sigma\left(\begin{pmatrix} \tilde{D}+I & 0 \\ 0 & \tilde{D}+I \end{pmatrix}^{-\frac{1}{2}} \begin{pmatrix} \tilde{A} & I \\ I & \tilde{A} \end{pmatrix} \begin{pmatrix} \tilde{D}+I & 0 \\ 0 & \tilde{D}+I \end{pmatrix}^{-\frac{1}{2}} \begin{pmatrix} H_n^{(l-1)} W^{(l)} \\ H_{ns}^{(l-1)} W^{(l)} \end{pmatrix} \right) \\
&= \sigma\left(\begin{pmatrix} (\tilde{D}+I)^{-\frac{1}{2}} \tilde{A} (\tilde{D}+I)^{-\frac{1}{2}} & \tilde{D}+I \\ \tilde{D}+I & (\tilde{D}+I)^{-\frac{1}{2}} \tilde{A} (\tilde{D}+I)^{-\frac{1}{2}} \end{pmatrix} \begin{pmatrix} H_n^{(l-1)} W^{(l)} \\ H_{ns}^{(l-1)} W^{(l)} \end{pmatrix} \right) \\
&= \sigma \begin{pmatrix} (\tilde{D}+I)^{-\frac{1}{2}} \tilde{A} (\tilde{D}+I)^{-\frac{1}{2}} H_n^{(l-1)} W^{(l)} + (\tilde{D}+I) H_{ns}^{(l-1)} W^{(l)} \\ (\tilde{D}+I)^{-\frac{1}{2}} \tilde{A} (\tilde{D}+I)^{-\frac{1}{2}} H_{ns}^{(l-1)} W^{(l)} + (\tilde{D}+I) H_n^{(l-1)} W^{(l)} \end{pmatrix},
\end{aligned} \tag{3}$$

where $\tilde{A} = A + I$, $\tilde{D} = D + I$ and $\tilde{D}^{'} = D^{'} + I$. $H_n^{(l-1)}$ and $H_{ns}^{(l-1)}$ is the node representation matrices of G and $\hat{G}$ after the $(l-1)$-th convolutional layer. The feature input of the 0_{th} layer is node feature matrices X^n and X^{ns}, which stack $\mathbf{x}_i^n$ and $\mathbf{x}_i^{ns}$ $(v_i \in G)$ across all graph nodes, respectively. $H^{(l)}$ is the node representation matrix of all nodes after the l_{th} convolutional layer. $W^{(l)}$ is the parameter matrix of the l_{th} convolutional layer. σ is a non-linear activation function.

Since the original node features and augmented node structural features can be very different, we employ two different convolutional filters (i.e., with different convolutional weights) to learn their knowledge as follows:

$$H^{(l)} = \begin{pmatrix} H_n^{(l)} \\ H_{ns}^{(l)} \end{pmatrix} \approx \sigma \begin{pmatrix} (\tilde{D}+I)^{-\frac{1}{2}}\tilde{A}(\tilde{D}+I)^{-\frac{1}{2}} H_n^{(l-1)} W_n^{(l)} + (\tilde{D}+I) H_{ns}^{(l-1)} W_{ns}^{(l)} \\ (\tilde{D}+I)^{-\frac{1}{2}}\tilde{A}(\tilde{D}+I)^{-\frac{1}{2}} H_{ns}^{(l-1)} W_{ns}^{(l)} + (\tilde{D}+I) H_n^{(l-1)} W_n^{(l)} \end{pmatrix}, \tag{4}$$

where $W_n^{(l)}$ and $W_{ns}^{(l)}$ are the parameter matrices of l_{th} layer for two types of features respectively, and $H_n^{(l)}$ and $H_{ns}^{(l)}$ are the node representation matrices of G and $\hat{G}$ after current l_{th} message passing layer.

After L message-passing layers, we aggregate the node representations of two graphs G and $\hat{G}$ in each layer to obtain the final node representation matrix as follows:

$$H = \mathrm{AGGATE}_n(H_n^{(1)}, \cdots, H_n^{(L)}, H_{ns}^{(1)}, \cdots, H_{ns}^{(L)}), \tag{5}$$

where $\mathrm{AGGATE}_n(\cdot)$ is an aggregate function, and concatenation is used in our experiments; H denotes the representation matrix that encapsulates the representation of all individual nodes. Then a readout function is applied to obtain the learned graph representation $\mathbf{h}_l$.

4.2.2. Message Passing in CoS-GIN

The framework can also be extended to other GNN backbones. Here we now present how the proposed message passing method can be adopted to the case using GIN as our backbone. To this end, the GIN-based message passing is re-defined as follows:

$$\begin{aligned}
\mathbf{h}_{v_i,n}^{(l)} &= \text{MLP}_n^{(l)}\left((1+\epsilon^{(l)})\mathbf{h}_{v_i,n}^{(l-1)} + \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{h}_{v_j,n}^{(l-1)} + \mathbf{h}_{v_i,ns}^{(l-1)}\right), \\
\mathbf{h}_{v_i,ns}^{(l)} &= \text{MLP}_{ns}^{(l)}\left((1+\epsilon^{(l)})\mathbf{h}_{v_i,ns}^{(l-1)} + \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{h}_{v_j,ns}^{(l-1)} + \mathbf{h}_{v_i,n}^{(l-1)}\right),
\end{aligned} \tag{6}$$

where MLP is a multi-layer perceptron layer. Then we combine the obtained representations via summation. In detail,

$$\begin{aligned}
\mathbf{h}_n &= \sum_l \text{FC}_n^{(l)}(\text{READOUT}(H_n^{(l)})), \\
\mathbf{h}_{ns} &= \sum_l \text{FC}_{ns}^{(l)}(\text{READOUT}(H_{ns}^{(l)})),
\end{aligned} \tag{7}$$

where $\text{FC}_n^{(l)}(\cdot)$ and $\text{FC}_{ns}^{(l)}(\cdot)$ are fully-connected layers in the l_{th} layer. We gain the learned graph representation $\mathbf{h}_l$ through adding them together:

$$\mathbf{h}_l = \mathbf{h}_n + \mathbf{h}_{ns}. \tag{8}$$

The key insight of the message passing mechanism in CoS-GIN is analogous to that in CoS-GCN, but they are derived at different representation levels: matrix of node representations in CoS-GCN vs. vectorized node representations in CoS-GIN, which is mainly done for presentation brevity.

4.3. Graph-level Representation Fusion

After gaining the learned graph representations $\mathbf{h}_l$, we then employ MLP to synthesize it, together with the augmented graph-structural feature $\mathbf{x}^{gs}$, to learn the final graph representations. In detail, we input $\mathbf{h}_l$ and $\mathbf{x}^{gs}$ into the two different MLPs as:

$$\mathbf{h}_l^{MLP} = \text{MLP}^l(\mathbf{h}_l), \mathbf{h}_{gs}^{MLP} = \text{MLP}^{gs}(\mathbf{x}^{gs}), \tag{9}$$

where $\text{MLP}^l(\cdot)$ and $\text{MLP}^{gs}(\cdot)$ are MLP functions. We then integrate the information learned to gain the final graph representation:

$$\mathbf{h}_g = \text{AGGATE}_g(\mathbf{h}_l^{MLP}, \mathbf{h}_{gs}^{MLP}), \tag{10}$$

where $\text{AGGATE}_g(\cdot)$ is the aggregation function and we use concatenation in our model. Then the graph representation can be used for any down-stream tasks. Algorithm 1 presents the procedure of CoS-GCN to calculate graph representations, which can be later input to any down-stream tasks.

Algorithm 1 Graph representation learning via CoS-GCN

```
Input: Graph set $\mathcal{G} = \{G_i\}_i$, two GNNs with parameter set $\{W_n^{(1)}, ..., W_n^{(L)}\}$
    and $\{W_{ns}^{(1)}, ..., W_{ns}^{(L)}\}$, two MLP functions $MLP^l(\cdot)$ and $MLP^{gs}(\cdot)$
Output: Graph representation $\mathbf{h}_g$ for $G \in \mathcal{G}$
Augment node and graph structural knowledge to obtain $X^{ns}$ and $\mathbf{x}^{gs}$ for
    each $G \in \mathcal{G}$
for $G$ in $\mathcal{G}$ do
    Compute $H^{(l)}, l \in \{1, \cdots, L\}$ with Eq.(4)
    Aggregate $H^{(l)}, l \in \{1, \cdots, L\}$ with Eq.(5) to obtain $H$
    Readout $H$ to obtain $\mathbf{h}_l$
    Input $\mathbf{h}_l$ and $\mathbf{x}^{gs}$ into $MLP^l$ and $MLP^{gs}$ respectively to gain $\mathbf{h}_l^{MLP}$ and
    $\mathbf{h}_{gs}^{MLP}$
    Aggregate $\mathbf{h}_l^{MLP}$ and $\mathbf{h}_{gs}^{MLP}$ to obtain the final representation $\mathbf{h}_g$ for $G$
end for
return Graph representation $\mathbf{h}_g$ for $G \in \mathcal{G}$
```

4.4. Expressive Power of CoS-GNN

This section discusses the expressive power of CoS-GNN. When comparing the expressiveness of GNN models, we can define that:

Definition 1. *For any two GNN models: A and B, model A is said to be more expressive than model B, if and only if 1) model A can distinguish all samples that model B can distinguish, and 2) there exists samples which can be distinguished by model A but not by model B.*

To measure the expressive power of GNNs, the Weisfeiler-Lehman (WL) graph isomorphism test is commonly used, which is a family of algorithms (k-

WL, k-FWL) used to test graph isomorphism [33, 34]. Two graphs G_1 and G_2 are called isomorphic if there exists an edge and color preserving bijection $\phi : \mathcal{V}_1 \to \mathcal{V}_2$. Next we show the strong expressive power of our model CoS-GNN from the WL-test perspective:

Theorem 1. *CoS-GNN is not less expressive than 1-WL and 2-WL tests.*

Proof. We first consider the comparison with 1-WL test. This equals to prove such statement: If CoS-GNN deems that two graphs are isomorphic, then 1-WL test will also deem them isomorphic. If after k iterations, the CoS-GNN regards two graphs G_1 and G_2 are isomorphic, we have $\mathbf{h}_{1,g}^{(k)} = \mathbf{h}_{2,g}^{(k)}$. Assuming that the $\mathrm{AGGATE_g}$ is injective, we can obtain that $\mathbf{h}_{1,l}^{MLP(k)} = \mathbf{h}_{2,l}^{MLP(k)}$ and $\mathbf{h}_{1,gs}^{MLP(k)} = \mathbf{h}_{2,gs}^{MLP(k)}$, followed by $\mathbf{h}_{1,l}^{(k)} = \mathbf{h}_{2,l}^{(k)}$ and $\mathbf{x}_1^{gs} = \mathbf{x}_2^{gs}$. Thus we have $H_1^{(i)} = H_2^{(i)}$ and then $\mathbf{h}_{v,n}^{(i)} = \mathbf{h}_{u,n}^{(i)}$ and $\mathbf{h}_{v,ns}^{(i)} = \mathbf{h}_{u,ns}^{(i)}$ for $v \in \mathcal{V}_{G_1}$, $u \in \mathcal{V}_{G_2}$ and $i = 1, ..., k$ when the $\mathrm{AGGATE_n}$ is injective.

What we need to prove next is that the color extracted by 1-WL for node v and u is same, $i.e. c_v^{(k)} = c_u^{(k)}$. We use the induction as [22] to demonstrate this. For $i = 0$, since the initial node features are the same for both CoS-GNN and 1-WL, we can get $c_v^{(0)} = c_u^{(0)}$ when $\mathbf{h}_{v,n}^{(0)} = \mathbf{h}_{u,n}^{(0)}$. Suppose $\mathbf{h}_{v,n}^{(j)} = \mathbf{h}_{u,n}^{(j)}, \mathbf{h}_{v,ns}^{(j)} = \mathbf{h}_{u,ns}^{(j)} \Rightarrow c_v^{(j)} = c_u^{(j)}$ holds for $j = 1, \cdots, k-1$, we later need to prove that it holds for $j = k$. Since each node representation, including $\mathbf{h}_{v,n}^{(j)}$ and $\mathbf{h}_{v,ns}^{(j)}$, is calculated by a COM function, if COM is injective, we have $\mathbf{h}_{v,n}^{(k-1)} = \mathbf{h}_{u,n}^{(k-1)}$, $\mathbf{h}_{v,ns}^{(k-1)} = \mathbf{h}_{u,ns}^{(k-1)}$, $\mathrm{AGGATE}(\{\mathbf{h}_{q,n}^{(k-1)} | q \in \mathcal{N}_v\}) = \mathrm{AGGATE}(\{\mathbf{h}_{p,n}^{(k-1)} | p \in \mathcal{N}_u\})$ and $\mathrm{AGGATE}(\{\mathbf{h}_{q,ns}^{(k-1)} | q \in \mathcal{N}_v\}) = \mathrm{AGGATE}(\{\mathbf{h}_{p,ns}^{(k-1)} | p \in \mathcal{N}_u\})$ when $\mathbf{h}_{v,n}^{(k)} = \mathbf{h}_{u,n}^{(k)}$ and $\mathbf{h}_{v,ns}^{(k)} = \mathbf{h}_{u,ns}^{(k)}$. According to Lemma 5 from [18], there exists an injective function. When AGGATE is injective, we have $\mathbf{h}_{q,n}^{(k-1)} = \mathbf{h}_{p,n}^{(k-1)}$ and $\mathbf{h}_{q,ns}^{(k-1)} = \mathbf{h}_{p,ns}^{(k-1)}$, which lead to $c_q^{(k-1)} = c_p^{(k-1)}$ for $q \in \mathcal{N}_v$ and $p \in \mathcal{N}_u$. Since we have $c_u^{(k-1)} = c_v^{(k-1)}$ according to the induction hypothesis, we can get $c_u^{(k)} = c_v^{(k)}$. Therefore, the 1-WL test regards two graphs isomorphic if the CoS-GNN regards them isomorphic.

Since 1-WL and 2-WL test have equivalent discrimination power [33, 25], CoS-GNN is also at least as expressive as 2-WL test. □

The theorem states that CoS-GNN is at least as expressive as 1-WL and 2-WL tests. Some graphs that 1-WL and 2-WL tests cannot distinguish can be identified by our CoS-GNN. For example, 1-WL and 2-WL fail to distinguish the two graphs in Figure 1, whereas CoS-GNN can easily differentiate them with the augmented features. Thus, our CoS-GNN can often learn more expressive representations than popular GNNs since they are mainly based on the 1-WL test, when handling complex graph datasets. For example, Chen et al. [35] have shown that MPNNs cannot perform induced-subgraph-count of any connected pattern consisting of 3 or more nodes. For graphs with subgraphs that MPNNs cannot learn to count, there would be some pairs of graphs with different number of such uncounted subgraphs that are regarded as isomorphic by MPNNs. On the other hand, CoS-GNN can discriminate these graphs through including structural features that differentiate these subgraphs. As shown in Figure 1, the two graphs cannot be distinguished by MPNNs, but they can be differentiated by the triangle counting for both nodes and graphs, and the existence of bridge in the graphs as well.

When compared with higher-order WL tests, we can also observe that our CoS-GNN can distinguish graphs that 2-FWL test (which is equivalent to 3-WL test [33]) fails to identify, meaning that 3-WL test is not more expressive than our CoS-GNN. For example, Arvind et al. [36] and Bouritsas et al. [22] have shown that the 2-FWL test fails to distinguish the well-known Rook's 4×4 and Shrikhande graphs, as illustrated in Figure 4. However, the clique features incorporated into our CoS-GNN model help effectively discriminate these two graphs.

4.5. Time Complexity Analysis

In this section, we analyze the time complexity of CoS-GNN. The computation cost mostly concentrates on the feature extraction stage and the message passing stage. Let n and m be the number of nodes and edges in the graph respectively, in the feature learning phase, the degree and triangle counting cost are $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ time respectively. The complexity of clique and core finding

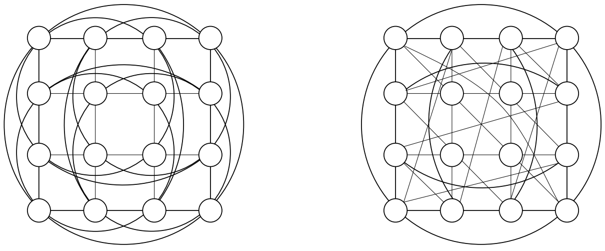


Figure 4: The strongly regular Rook's 4×4 graph (left) and Shrikhande graph (right) [22, 36]. The 3-WL/2-FWL test is not able to deem them as non-isomorphic. Rook's 4×4 graph possesses 4-cliques while the Shrikhande graph features 5-rings, which are not present in Rook's.

are respectively bounded by $\mathcal{O}(n{*}3^n)$ and $\mathcal{O}(n{+}m)$. The computation of triangle and square clustering coefficient is $\mathcal{O}(n^2)$. The bridge finding needs $\mathcal{O}(n{+}m)$ time. The average clustering coefficient, average global and local efficiency require $\mathcal{O}(n^2)$, $\mathcal{O}(n^3)$ and $\mathcal{O}(n^4)$ respectively. Therefore, the feature extraction stage requires $\mathcal{O}(n^4{+}n{*}3^n{+}m)$ time. As for the message passing stage, the time complexity of our CoS-GNN equals to the corresponding vanilla GNN. Thus, the total time complexity of our CoS-GNN is $\mathcal{O}(n^4 + n * 3^n + m) + \mathcal{O}_{GNN}$.

5. Experiments and Results

5.1. Datasets

We perform experiments on 12 publicly available datasets from the TUDataset graph classification benchmark [45] to justify the effectiveness of our CoS-GNN. The detailed information of the datasets is displayed in Table 1.

5.2. Competing Methods and Evaluation Metrics

Our method CoS-GNN is compared with 13 state-of-the-art (SOTA) methods:

- **Graph kernels.** We use two graph kernels, *i.e.***Weisfeiler-lehman subtree kernel (WL)** [46] and **Propagation graph kernels (PK)** [47] as baselines.

Table 1: The detailed information of 12 public datasets. The following acronyms, PROTEINS_full (PROTS_full), IMDB-BINARY (I-BINARY), IMDB-MULTI (I-MULTI), REDDIT-BINARY (R-BINARY) and REDDIT-MULTI-5K (R-MULTI), are used. The 'binary' in the 'Class' column denotes the dataset is for binary classification while 'multi' implies multi-class classification. The '#Graphs' is the total number of graphs in the dataset and the '#Nodes' means the average number of nodes in the dataset. The '✓' in the 'Attribute' column indicates the data contains attributed graphs, and otherwise they contain only plain graphs.

Dataset	**Area**	**Class**	**#Graphs**	**#Nodes**	**Attribute**
BZR	molecule	binary	405	35.75	✓
COX2	molecule	binary	467	41.22	✓
DD	bioinformatics	binary	1178	284.32	-
I−BINARY	social	binary	1000	19.77	-
I−MULTI	social	multi	1500	13.00	-
MUTAG	molecule	binary	188	17.93	-
NCI1	molecule	binary	4110	29.87	-
NCI109	molecule	binary	4127	29.68	-
PROTS_full	bioinformatics	binary	1113	39.06	✓
R−BINARY	social	binary	2000	429.63	-
R−MULTI	social	multi	4999	508.52	-
ENZYMES	bioinformatics	multi	600	32.63	✓

- **Basic graph neural networks.** We consider four popular networks, *i.e.*, **GCN** [15], **SAGE** [16], **GAT** [17] and **GIN** [18], as the network baselines.

- **GNN-based augmentation methods.** We also compare CoS-GNN with several augmentation models that are built based on GNNs, including $\mathcal{G}$**-mixup** [30], **Dummy** [29], **DropGNN** [32], **rGIN** [28], **NestedGNN** [25], **LAGNN** [24], and **GSN** [22].

In terms of performance evaluation, we employ accuracy and Area Under Precision-Recall Curve (AUPRC) as the evaluation metrics for graph classification while Area Under Receiver Operating Characteristic Curve (AUC) for anomaly detection. Higher accuracy/AUPRC/AUC indicates better performance. We report the mean results and standard deviation based on 10-fold cross-validation for all datasets.

5.3. Implementation Details

All experiments are executed on NVIDIA Quadro RTX 6000 GPU with an Intel Xeon E-2288G 3.7GHz CPU, and all models are implemented with Python 3.8[1]. The following parameters are set by default for CoS-GCN and its competing methods, including WL, PK, GAT, SAGE and GCN, on all 12 datasets: the learning rate is 0.001, the batch size is set to 512, the number of network layers is 3, the hidden layer dimension of network is 256, the classifier is a 3-layer MLP, pooling operation is max pooling, and the number of epochs is 1,000. The iteration number of WL is 3. For GIN and CoS-GIN, the learning rate is chosen from $\{0.01, 0.001, 0.0005, 0.0001\}$, the batch size is selected from $\{32, 64, 128, 256\}$, hidden layer dimension is ranged in $\{16, 64, 128, 256\}$ and the readout operation is either meanpooling or maxpooling. For other baselines, we run their public codes with their recommended settings.

[1] https://www.python.org/

Table 2: Accuracy (mean±std) of CoS-GNN and SOTA competing methods for graph classification on 12 real-world datasets. The best and second performance per dataset is boldfaced and underlined respectively. The following acronyms, PROTEINS_full (PROTS_full), IMDB-BINARY (I-BINARY), IMDB-MULTI (I-MULTI), REDDIT-BINARY (R-BINARY) and REDDIT-MULTI-5K (R-MULTI), are used. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance.

Method	BZR	COX2	DD	I-BINARY	I-MULTI	MUTAG	NCI1	NCI109	PROTS_full	R-BINARY	R-MULTI	ENZYMES	Rank	p-value
WL	0.790±0.045	0.760±0.043	0.747±0.023	0.727±0.035	0.503±0.021	0.797±0.072	0.643±0.025	0.647±0.021	0.737±0.028	0.611±0.039	0.382±0.019	0.338±0.074	13.5	0.0005
PK	0.818±0.026	0.790±0.015	0.742±0.026	0.729±0.050	0.488±0.038	0.697±0.054	0.634±0.017	0.620±0.036	0.734±0.026	0.635±0.049	0.411±0.024	0.392±0.036	14.3	0.0005
GCN	0.840±0.028	0.811±0.043	0.756±0.033	0.724±0.036	0.498±0.024	0.744±0.102	0.785±0.016	0.769±0.026	0.749±0.028	0.900±0.024	0.533±0.013	0.450±0.068	8.8	0.0005
GAT	0.837±0.036	0.790±0.065	0.773±0.027	0.704±0.044	0.496±0.035	0.738±0.105	0.772±0.015	0.765±0.027	0.748±0.035	0.851±0.036	0.502±0.025	0.422±0.069	10.8	0.0010
SAGE	0.842±0.055	0.820±0.055	0.771±0.031	0.733±0.046	0.495±0.024	0.766±0.059	0.795±0.018	0.780±0.027	0.748±0.034	0.878±0.029	0.508±0.017	0.395±0.079	8.3	0.0005
GIN	0.835±0.033	0.805±0.039	0.750±0.048	0.731±0.047	0.502±0.031	0.866±0.078	0.790±0.023	0.795±0.012	0.738±0.039	0.891±0.016	0.557±0.021	0.550±0.091	7.3	0.0005
$\mathcal{G}$**−mixup**	0.832±0.042	0.805±0.043	–	0.719±0.030	0.505±0.015	0.824±0.093	0.778±0.023	0.752±0.039	0.705±0.054	**0.929**±0.009	0.555±0.005	0.487±0.054	8.9	0.0049
Dummy	0.828±0.069	0.805±0.048	0.777±0.035	–	–	0.829±0.105	0.752±0.015	0.738±0.025	**0.770**±0.029	–	–	0.500±0.101	8.4	0.0034
DropGNN	0.845±0.030	0.807±0.055	–	0.741±0.027	0.502±0.023	0.834±0.095	0.809±0.014	0.810±0.013	0.747±0.032	–	–	0.588±0.053	5.7	0.0005
rGIN	0.827±0.046	0.805±0.051	0.698±0.020	0.746±0.022	0.512±0.026	0.851±0.052	0.808±0.019	0.801±0.016	0.757±0.024	0.897±0.020	0.549±0.023	0.663±0.073	6.0	0.0098
NestedGCN	0.808±0.033	0.782±0.009	0.763±0.038	0.733±0.052	0.499±0.037	0.829±0.011	0.720±0.080	0.708±0.076	0.730±0.017	–	–	0.312±0.067	12.8	0.0005
NestedGIN	0.832±0.082	0.790±0.066	**0.778**±0.039	0.745±0.064	0.510±0.023	0.879±0.082	0.777±0.017	0.779±0.026	0.738±0.035	–	–	0.290±0.080	7.8	0.0015
GSN-v	0.825±0.036	0.801±0.040	0.702±0.033	**0.760**±0.047	0.509±0.031	0.861±0.097	0.808±0.018	0.804±0.020	0.736±0.036	0.819±0.028	0.521±0.026	**0.687**±0.043	7.6	0.0190
LAGCN	0.837±0.039	0.807±0.064	0.752±0.033	0.727±0.041	0.499±0.022	0.761±0.087	0.752±0.027	0.713±0.031	0.755±0.035	0.884±0.016	0.513±0.026	0.590±0.047	9.3	0.0010
LAGIN	0.820±0.035	**0.831**±0.028	0.766±0.037	0.731±0.055	0.491±0.032	0.872±0.049	0.763±0.016	0.740±0.032	0.760±0.032	0.894±0.020	0.553±0.022	0.645±0.042	7.6	0.0156
CoS-GCN	0.832±0.036	0.824±0.055	**0.778**±0.032	0.754±0.052	0.503±0.019	0.878±0.059	0.816±0.013	0.801±0.021	0.757±0.028	0.917±0.022	0.554±0.028	0.533±0.064	3.8	–
CoS-GIN	**0.850**±0.043	0.822±0.094	0.773±0.038	0.753±0.033	**0.517**±0.023	**0.883**±0.066	**0.823**±0.026	**0.812**±0.013	0.751±0.019	0.906±0.018	**0.559**±0.021	0.653±0.036	**2.3**	–

Table 3: AUPRC (mean±std) of CoS-GNN and SOTA competing methods for graph classification on 9 real-world binary classification datasets. The best and second performance per dataset is boldfaced and underlined respectively. The following acronyms, PROTEINS_full (PROTS_full), IMDB-BINARY (I-BINARY), IMDB-MULTI (I-MULTI), REDDIT-BINARY (R-BINARY) and REDDIT-MULTI-5K (R-MULTI), are used. 'Rank' indicates the average performance ranking of a model across all datasets: a smaller rank value indicates a better overall performance.

Method	BZR	COX2	DD	I-BINARY	MUTAG	NCI1	NCI109	PROTS_full	R-BINARY	Rank
GCN	0.674±0.124	0.541±0.152	0.760±0.055	0.831±0.038	0.783±0.113	0.842±0.023	0.824±0.0198	0.763±0.038	0.956±0.011	7.3
GAT	0.624±0.156	0.520±0.137	0.753±0.058	0.814±0.042	0.870±0.089	0.841±0.020	0.811±0.014	**0.772**±0.038	0.937±0.022	9.6
SAGE	0.661±0.170	0.565±0.144	**0.803**±0.038	0.813±0.042	0.899±0.0569	0.852±0.029	0.823±0.0176	0.760±0.041	0.944±0.015	6.8
GIN	0.619±0.123	0.556±0.115	0.770±0.037	0.828±0.032	0.963±0.026	0.857±0.028	0.848±0.021	0.699±0.058	0.940±0.034	7.3
$\mathcal{G}$−mixup	0.691±0.130	0.439±0.160	–	0.829±0.033	0.945±0.040	0.833±0.021	0.796±0.043	0.688±0.078	–	10.6
Dummy	0.570±0.067	0.554±0.175	0.800±0.040	–	0.915±0.074	0.808±0.031	0.786±0.027	**0.772**±0.040	–	9.6
DropGNN	0.675±0.085	0.555±0.172	0.690±0.044	0.837±0.024	0.962±0.027	**0.879**±0.020	0.863±0.021	0.708±0.051	0.938±0.029	6.1
rGIN	0.576±0.096	0.573±0.147	0.704±0.026	0.821±0.022	0.968±0.025	0.868±0.020	0.862±0.010	0.706±0.068	0.928±0.039	7.7
NestedGIN	0.587±0.233	0.563±0.180	0.784±0.049	**0.840**±0.039	0.962±0.029	0.851±0.020	0.842±0.025	0.745±0.050	–	6.9
GSN-v	0.694±0.143	0.528±0.150	0.699±0.050	0.828±0.073	0.969±0.031	0.857±0.0215	0.847±0.021	0.670±0.132	0.876±0.037	8.2
LAGCN	0.652±0.077	0.560±0.113	0.757±0.045	0.826±0.038	0.912±0.034	0.800±0.020	0.755±0.026	0.747±0.040	0.961±0.025	8.8
LAGIN	0.552±0.089	0.539±0.130	0.785±0.066	0.833±0.025	**0.979**±0.012	0.838±0.024	0.811±0.044	0.723±0.067	0.955±0.020	7.7
CoS-GCN	0.594±0.102	**0.575**±0.133	0.795±0.052	0.837±0.046	0.974±0.020	**0.879**±0.014	0.861±0.027	0.760±0.039	0.960±0.022	**3.4**
CoS-GIN	**0.724**±0.147	0.562±0.160	0.777±0.046	0.829±0.034	0.977±0.019	0.872±0.015	**0.880**±0.021	0.725±0.035	**0.963**±0.011	3.7

5.4. Enabling Graph Classification

The graph classification accuracy results of CoS-GNN models (including CoS-GCN and CoS-GIN) and 12 SOTA competing methods are reported in Table 2, where the GNN backbone used in $\mathcal{G}$-mixup, Dummy and DropGNN is all GIN due to its better performance; the results of $\mathcal{G}$-mixup on the IMDB and REDDIT datasets are taken from [30]; the result of Dummy on DD, NCI1 and NCI109 are from [29]; the results of NestedGCN and NestedGIN on DD, MUTAG and ENZYMES are from [25]; and '-' means the results are not reported in the original papers.

It is clear that CoS-GIN and CoS-GCN achieve the best or second-best performance on most of the datasets and the two top-ranked methods among all methods. Specifically, CoS-GCN improves GCN by 0.8%, 2.2%, 3.0%, 3.1%, 3.2% and 8.3% for PROTEINS_full, DD, IMDB-BINARY, NCI1, NCI109 and ENZYMES respectively, while the improvements brought by CoS-GIN over GIN are 1.5%, 1.7%, 1.7%, 2.2%, 2.3%, 3.3% and 10.3% for BZR, COX2, IMDB-BINARY, DD, NCI1, NCI109 and ENZYMES respectively. These large performance advancement reveals that the structural information in these dataset is specific and the feature augmentation and message passing process in our CoS-GNN makes full use of these structural information to improve its performance. When compared with other augmentation methods, our models can also perform better than the SOTA models on most datasets (*i.e.*, NCI109 (0.2%), REDDIT-MULTI (0.2%), MUTAG (0.4%), BZR (0.5%), IMDB-MULTI (0.5%) and NCI1 (1.4%)) and ranks top among all the competitors on overall performance. We also perform a paired Wilcoxon signed rank test to examine the significance of CoS-GNN against each of the competing methods across the 12 datasets. As shown by the p-values in Table 2, our CoS-GIN significantly outperforms GSN-v and LAGIN at the 95% confidence level and exceeds other competitors at the 99% confidence level. These results indicate that our collective node and graph structural knowledge augmented GNNs can learn more important graph structure information for graph classification. Besides, on individual datasets, CoS-GNN can gain 2%-11% accuracy improvement maximally on specific datasets when compared to the best-performing competing methods NestedGNN, GSN-v and LAGNN (for example, 5% enhancement of NestedGIN on NCI1, 9% improvement of GSN on REDDIT-BINARY). This means that the domain-adaptive graph structural knowledge in CoS-GNN can provide more generalized information to improve the model performance across different datasets while NestedGNN, GSN-v and LAGNN only consider the local structural information, which limits their performance. In summary, compared to each SOTA method, CoS-GNN may only have limited improvements on a few individual datasets, but the improvement on a set of datasets is substantial, and its improvement is significant across the 12 datasets used.

We report the AUPRC results of CoS-GNN and the competing methods on binary classification tasks in Table 3. Considering the limited performance of WL, PK and LAGCN, we omit their results. As can be seen in Table 3, although our CoS-GNN is not always the best model on every dataset, our CoS-GIN and CoS-GCN still achieve the top two performance on overall datasets, which further demonstrates the excellent ability of our CoS-GNN.

We also compare our CoS-GNN with vanilla GNNs on Open Graph Benchmark (OGB) dataset–ogbg-molhiv and ogbg-molpcba in Table 4. Our CoS-GNN achieves better performance than corresponding vanilla GNN in most situations, indicating the positive contribution of the augmented features. The performance of CoS-GIN is a bit worse than that of GIN on ogbg-molpcba, which might be because that although our augmented features are useful, which is demonstrated by the improvement of CoS-GCN compared with GCN, GIN has also learned enough useful structural information and the augmentation operation in our CoS-GIN does not provide extra discriminative information.

We also calculate the training and inference time of our CoS-GNN and its competitors to demonstrate the efficiency of the CoS-GNN. We use the same GIN structure in all models. The results are reported in Table 5. We can see that our CoS-GIN is a little more costly than simple augmentation operation with conventional GIN module, which is caused by the feature augmentation operation, but it is more efficient than complex structural augmentation meth-

Table 4: Results (mean±std) of CoS-GNN and corresponding vanilla GNN on OGB datasets – ogbg-molhiv and ogbg-molpcba. The best performance per dataset is boldfaced.

	ogbg-molhiv	ogbg-molpcba
Model	**AUROC**	**AP**
GCN	0.7626±0.0098	0.1753±0.0023
CoS-GCN	**0.7662 ± 0.0165**	**0.2045 ± 0.0034**
GIN	0.7825±0.0077	**0.2288 ± 0.0027**
CoS-GIN	**0.7912 ± 0.0068**	0.2249±0.0034

ods, including DropGNN and NestedGIN. Besides, although our CoS-GIN is a little time-costly on some large-scale datasets, it can still be successfully implemented on devices with limited computational ability, while $\mathcal{G}$-mixup, dummy, DropGNN and NestedGIN require more powerful devices on such instances, which restricts their application.

Table 5: Training and inference time of augmentation methods in graph classification task. All methods are with GIN as backbone. Each result is the time on the whole dataset.

Stage	**Method**	**BZR**	**COX2**	**DD**	**I-BINARY**	**I-MULTI**	**MUTAG**	**NCI1**	**NCI109**	**PROTS_full**	**R-BINARY**	**R-MULTI**	**ENZYMES**
Training	$\mathcal{G}$-mixup	0.0281	0.0328	-	0.0548	0.0679	0.0107	0.2547	0.2027	0.0684	-	-	0.0369
	Dummy	0.0186	0.0218	0.2356	-	-	0.0095	0.1556	0.1527	0.0468	-	-	0.0230
	DropGNN	0.3380	0.2419	-	0.2009	0.4198	0.0543	2.2575	2.3114	1.1480	-	-	0.4901
	rGIN	0.0482	0.0558	0.3594	0.1035	0.1231	0.0213	0.4054	0.3735	0.1172	0.8252	2.1535	0.0590
	NestedGIN	0.1861	0.2056	9.3848	0.8087	0.6975	0.0545	1.3730	1.4143	0.6841	-	-	0.2964
	GSN-v	1.7383	1.9700	6.1113	4.6320	3.3124	1.1959	1.3851	1.3868	1.3944	39.6010	50.8742	1.3267
	LAGIN	0.0492	0.0550	0.3569	0.1009	0.1167	0.0215	0.3647	0.3670	0.1167	0.7567	2.0873	0.0596
	CoS-GIN	0.0870	0.0991	0.7040	0.1719	0.1978	0.0358	0.5852	0.5855	0.1994	1.5373	4.3893	0.1020

Stage	**Method**	**BZR**	**COX2**	**DD**	**I-BINARY**	**I-MULTI**	**MUTAG**	**NCI1**	**NCI109**	**PROTS_full**	**R-BINARY**	**R-MULTI**	**ENZYMES**
Inference	$\mathcal{G}$-mixup	0.0029	0.0029	-	0.0042	0.0054	0.0023	0.0122	0.0125	0.0049	-	-	0.0033
	Dummy	0.0029	0.0029	0.0186	-	-	0.0022	0.0117	0.0120	0.0050	-	-	0.0034
	DropGNN	0.0188	0.0132	-	0.0116	0.0184	0.0061	0.1079	0.1161	0.0533	-	-	0.0247
	rGIN	0.0053	0.0054	0.0218	0.0079	0.0099	0.0049	0.0227	0.0216	0.0086	0.0376	0.1185	0.0056
	NestedGIN	0.0147	0.0148	0.5884	0.0632	0.0529	0.0056	0.0909	0.0936	0.0404	-	-	0.0207
	GSN-v	0.0088	0.0095	0.0306	0.0160	0.0153	0.0082	0.0246	0.0250	0.0095	0.1121	0.3701	0.0081
	LAGIN	0.0032	0.0035	0.0168	0.0046	0.0045	0.0035	0.0046	0.0046	0.0051	0.0286	0.0478	0.0037
	CoS-GIN	0.0092	0.0094	0.0410	0.0116	0.0132	0.0079	0.0321	0.0324	0.0129	0.0807	0.2334	0.0101

5.5. Employ CoS-GNN as GNN Backbone

5.5.1. Combined with GNN-based Methods

In this section, we examine the applicability of our CoS-GNN as GNN backbone in other GNN-based methods by replacing the GCN of $\mathcal{G}$-mixup and Dummy with CoS-GCN. We omit the results on REDDIT-BINARY and REDDIT-MULTI here because we can not run $\mathcal{G}$-mixup and Dummy on them by our device. The accuracy results of GCN-based and CoS-GCN-based $\mathcal{G}$-mixup and Dummy are reported in Table 6. The results show that the CoS-GCN-based $\mathcal{G}$-mixup outperform GCN-based $\mathcal{G}$-mixup on all datasets and the largest improvement can be up to 32%. The performance of CoS-GCN-based Dummy method is also better than the GCN-based Dummy on most datasets, achieving up to 25% improvement. The paired signed-rank test indicates that the improvement of CoS-GCN-based $\mathcal{G}$-mixup and Dummy across 10 datasets is significant at 99% and 90% confidence level, respectively. The decrease in accuracy of CoS-GCN-based Dummy on DD and PROTEINS_full might be because that the specific structural statistics augmented on the graphs are influenced and disturbed by the addition of the dummy node. When compared with the results of single CoS-GCN, there are also some improvement brought by CoS-GCN-based $\mathcal{G}$-mixup on BZR (3.5%), MUTAG (0.5%) and NCI1 (0.5%) and by CoS-GCN-based Dummy on ENZYMES (7.4%), which means that the combination with other GNN-based methods can also enhance the power of CoS-GNN. Overall, our CoS-GNN and other GNN-based methods are complementary and can be used as the basic GNN module in GNN-based methods to improve their performance successfully.

5.5.2. Combined with Other Pooling Methods

There have been a type of pooling methods that hierarchically extract the graph information [48, 49]. We demonstrate that our CoS-GNN structure can be combined with these methods in this section. In each pooling layer, we use the real node features to calculate the pooling criterion and construct the coarsened graph.

Table 6: Accuracy (mean±std) results of $\mathcal{G}$-mixup and Dummy using CoS-GCN as the GNN module, with $\mathcal{G}$-mixup and Dummy with GCN as baselines in graph classification. 'Different' denotes accuracy improvement (↑) or decrease (↓) brought by the replacement of CoS-GCN. Both of two methods suffer out of memory on REDDIT-BINARY and REDDIT-MULTI.

Dataset	**$\mathcal{G}-$mixup(GCN)**	**$\mathcal{G}-$mixup(CoS-GCN)**	**Difference**
BZR	0.8295 ± 0.0188	0.8666 ± 0.0471	0.0371 ↑
COX2	0.7730 ± 0.0482	0.7967 ± 0.0416	0.0237 ↑
DD	–	–	–
I−BINARY	0.7360 ± 0.0350	0.7410 ± 0.0243	0.0050 ↑
I−MULTI	0.5013 ± 0.0283	0.5073 ± 0.0264	0.0060 ↑
MUTAG	0.7173 ± 0.1130	0.8830 ± 0.0617	0.1657 ↑
NCI1	0.5007 ± 0.0010	0.8212 ± 0.0146	0.3205 ↑
NCI109	0.5038 ± 0.0007	0.7986 ± 0.0179	0.2948 ↑
PROTS_full	0.7152 ± 0.0350	0.7512 ± 0.0246	0.0360 ↑
ENZYMES	0.3456 ± 0.0435	0.4909 ± 0.0490	0.1453 ↑
p-value	0.0039	-	-

Dataset	**Dummy(GCN)**	**Dummy(CoS-GCN)**	**Difference**
BZR	0.8296 ± 0.0203	0.8321 ± 0.0482	0.0025 ↑
COX2	0.7899 ± 0.0509	0.8112 ± 0.0654	0.0213 ↑
DD	0.7776 ± 0.0717	0.7699 ± 0.0433	−0.0077 ↓
I−BINARY	–	–	–
I−MULTI	–	–	–
MUTAG	0.7813 ± 0.1292	0.8673 ± 0.0754	0.0860 ↑
NCI1	0.6608 ± 0.1016	0.8092 ± 0.0146	0.1484 ↑
NCI109	0.5527 ± 0.0851	0.8013 ± 0.0243	0.2486 ↑
PROTS_full	0.7557 ± 0.0375	0.7556 ± 0.0286	−0.0001 ↓
ENZYMES	0.4450 ± 0.1038	0.6067 ± 0.0602	0.1617 ↑
p-value	0.0547	-	-

We run our CoS-GCN with a hierarchical pooling–MVPool as the pooling operation to prove that our CoS-GCN can improve the performance of hierarchical pooling. The results of CoS-GCN with MVPool and GCN with MVPool as baseline are reported in Table 7, showing that CoS-GCN still can bring improvement on all datasets except COX2 and IMDB-MULTI. The average improvement is 2.02% and the maximal improvement can be up to about 12.28%. The paired signed-rank test indicates that the improvement across 12 datasets is significant at 99% confidence level. These results demonstrate that the augmented node and graph structural information also can provide extra useful information while the pooling operation is learning graph structural information. The accuracy decline of CoS-GCN with MVPool on COX2 is very marginal, only 0.01%. The 1% drop of CoS-GCN with MVPool on IMDB-MULTI might be because that the structural information in IMDB-MULTI might be limited and the hierarchical learning of MVPool can utilize most structural information in the graph. The feature augmentation in CoS-GCN provides some redundant information to the CoS-GCN-MVPool.

5.6. Enabling Other Down-stream Tasks

5.6.1. Graph Anomaly Detection

We next evaluate the performance of CoS-GNN in anomaly detection, in which the normal graph samples are available for training. It should be noted that this experiment is focused to demonstrate the ability of CoS-GNN in enabling better performance of some popular anomaly detection algorithms, compared to the use of original GNNs, rather than to argue for state-of-the-art anomaly detection performance of CoS-GNN. Thus, we use one-class GCN (OCGCN) and GLocalKD [50] as baselines and replace the GCN of GLocalKD with CoS-GCN to examine the ability of CoS-GCN to enable the anomaly detector GLocalKD. Since GLocalKD employs degree information as the node features for plain graphs, which is one of the features we augment in CoS-GNN, we only compare the performance of them on graphs with node attributes. The datasets we use in this experiment are all from TUDataset graph classification

Table 7: Accuracy (mean±std) results of GCN and CoS-GCN with MVPool as the readout operation. ‘Different’ denotes accuracy improvement (↑) or decrease (↓) brought by CoS-GCN compared to GCN.

Dataset	**GCN-MVPool**	**CoS-GCN-MVPool**	**Difference**
BZR	0.8273 ± 0.0565	0.8345 ± 0.0301	$0.0072 \uparrow$
COX2	0.7987 ± 0.0351	0.7986 ± 0.0430	$-0.0001 \downarrow$
DD	0.7750 ± 0.0363	0.7962 ± 0.0390	$0.0212 \uparrow$
I−BINARY	0.7280 ± 0.0268	0.7350 ± 0.0492	$0.0070 \uparrow$
I−MULTI	0.5180 ± 0.0253	0.5080 ± 0.0332	$-0.0100 \downarrow$
MUTAG	0.7178 ± 0.0858	0.8406 ± 0.1107	$0.1228 \uparrow$
NCI1	0.7791 ± 0.0155	0.8015 ± 0.0094	$0.0224 \uparrow$
NCI109	0.7754 ± 0.0233	0.7989 ± 0.0198	$0.0235 \uparrow$
PROTS_full	0.7556 ± 0.0360	0.7664 ± 0.0298	$0.0108 \uparrow$
R−BINARY	0.9050 ± 0.0219	0.9140 ± 0.0202	$0.0090 \uparrow$
R−MULTI	0.5311 ± 0.0136	0.5515 ± 0.0255	$0.0204 \uparrow$
ENZYMES	0.5833 ± 0.0516	0.5917 ± 0.0455	$0.0084 \uparrow$
p-value	0.0093	-	-

benchmark [45], with the results reported in Table 8, where FTEIN is short for FRANKENSTEIN. It can be observed that CoS-GCN improves the performance of GLocalKD largely on most datasets and the largest improvement can be up to 24.6%, which means that the augmented node and graph structural features can also be effectively leveraged via our proposed message passing for improving the detection of anomalies, despite it is an semi-supervised task. The decrease of CoS-GCN on AIDS might be because that AIDS is a rather simple dataset on which the original GLocalKD has obtained an AUC of almost one; CoS-GCN is slightly over-parameterized for such a simple dataset.

5.6.2. Out-of-distribution Generalization

We evaluate the generalization ability of CoS-GNN on out-of-distribution (OOD) data in this section. This experiment is designed to compare the performance of CoS-GNN with other two message passing neural networks (*i.e.*GCN and GIN) in OOD generalization. These GNNs can be utilized as GNN back-

Table 8: AUC results (mean±std) of OCGCN, GLocalKD based on GCN, and CoS-GCN-enbaled GLocalKD (CoS-GCN for short) on 12 public attributed graph datasets. 'Diff.' denotes AUC improvement (↑) or decrease (↓) resulted by replacing the GCN backbone with CoS-GCN in GLocalKD.

Dataset	**OCGCN**	**GLocalKD**	**CoS-GCN**	**Diff.**
AIDS	0.664 ± 0.080	0.992 ± 0.004	0.948 ± 0.008	−0.044 ↓
BZR	0.658 ± 0.071	0.679 ± 0.065	0.804 ± 0.068	0.125 ↑
COX2	0.628 ± 0.072	0.589 ± 0.045	0.665 ± 0.050	0.076 ↑
DHFR	0.495 ± 0.080	0.558 ± 0.030	0.595 ± 0.053	0.037 ↑
PROTS_full	0.718 ± 0.036	0.785 ± 0.034	0.792 ± 0.024	0.007 ↑
ENZYMES	0.613 ± 0.087	0.636 ± 0.061	0.760 ± 0.070	0.124 ↑
COIL−RAG	0.629 ± 0.210	0.656 ± 0.220	0.700 ± 0.082	0.044 ↑
Letter−high	0.580 ± 0.042	0.591 ± 0.023	0.655 ± 0.071	0.064 ↑
Letter−low	0.616 ± 0.168	0.738 ± 0.051	0.984 ± 0.005	0.246 ↑
Letter−med	0.618 ± 0.080	0.662 ± 0.062	0.852 ± 0.024	0.190 ↑
FTEIN	0.550 ± 0.031	0.547 ± 0.019	0.563 ± 0.018	0.016 ↑
Synthie	0.568 ± 0.083	0.844 ± 0.036	0.862 ± 0.017	0.018 ↑

bone in various generalization methods to obtain better performance further. The datasets we utilize are from GOOD[2] [51]. GOOD-Motif is a synthetic dataset designed for structure shifts, GOOD-HIV is a molecular dataset, and GOOD-SST2 is a natural language sentiment analysis dataset. For each dataset, the GOOD benchmark selects one or two domain features (*e.g.*, base and size for GOOD-Motif, scaffold and size for GOOD-HIV, and length for GOOD-SST2) and then applies covariate and concept shift splits per domain to create diverse distribution shifts. Following [51], the metric we use for GOOD-HIV is AUC and classification accuracy is used for other datasets. We examine the generalization power of CoS-GNN with the baseline models taken from the GOOD benchmark [51]. The GNN backbone used in the baselines is GIN.

The results on the OOD and the in-distribution (ID) validation sets are reported in Table 9. It can be seen from the results that our model CoS-GCN

[2] https://github.com/divelab/GOOD/

outperforms the basic GCN on all settings except the one on GOOD-HIV; CoS-GIN gains better performance than GOOD on all settings except GOOD-HIV. This is mainly because that the node and graph structures augmented in our CoS-GNN are more generalizable w.r.t. different shifts of base, size, or length on the three GOOD datasets, while being less generalizable to the scaffold shift, a two-dimensional structural base of a molecule. The especially outstanding performance of CoS-GNN on GOOD-Motif also helps justify this. Each graph in GOOD-Motif is generated by connecting a base graph and a motif, and thus, the structure of base graphs and motifs is highly differentiated. Thus, the augmented structural information of each class enables the structure learning in CoS-GNN to obtain substantially improved OOD generalization performance, when compared with vanilla GNN.

5.7. Robustness w.r.t. Structure Contamination

Since the data collected in real applications may be with limited/noisy structural information, the performance of our CoS-GNN, which harnesses rich structural information, might be influenced by these contaminated information. In this section, we discuss the impact of limited/noisy structural knowledge on our CoS-GNN. Specifically, we randomly remove $\{1\%, 5\%, 10\%, 15\%, 20\%\}$ edges of the data and compare their results with the results on original data. Our experiment is implemented on GCN backbone.

The results on NCI1 is displayed in Figure 5. It is obvious that both CoS-GCN and GCN suffer from performance decline due to the edge removal and the decline level of them is similar. Our CoS-GCN always have better performance than GCN under various structural contamination situation. This means that limited/noisy structure brings no more serious effects on the CoS-GCN. This might be because that the structures we augment are in different scales and parts of the extracted features will be infected while others will still be exact. The unaffected structure features can correct the influence of the wrong information brought by the limited/noisy graph structure.

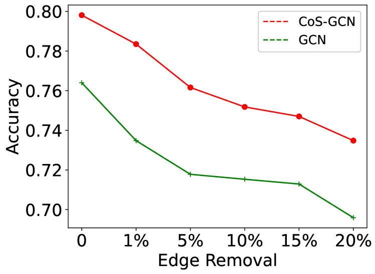


Figure 5: Accuracy performance of CoS-GCN and GCN w.r.t. different structural contamination rates.

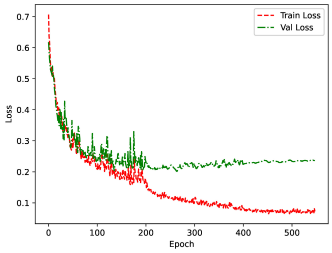


Figure 6: Loss variation tendency of CoS-GCN on the training and validation dataset of REDDIT-BINARY.

5.8. Convergence Analysis

In this section we run an experiment to illustrate the convergence ability of our CoS-GNN. In detail, we run the CoS-GCN on the REDDIT-BINARY dataset and record the loss tendency of training and validation dataset. The result is shown in Figure 6. It is obvious that both the training and validation loss will approach stability after a number of epochs. Besides, the early stopping used during training can ensure the model against overfitting and obtaining an excellent result.

5.9. Ablation Study

5.9.1. Ablation Study of the Specific Message Passing Scheme

This section examines the importance of the graph augmentation and the message passing scheme designed in CoS-GNN. All expeirments are based on

CoS-GCN. We first evaluate the performance of GCN with original/augmented features as sole input ($\mathcal{V}_{nf}$ for real node feature, $\mathcal{V}_{ns}$ for augmented node structure features, and $\mathcal{V}_{gs}$ for augmented graph structure features), and then combine original and augmented node features by convolution after concatenation ($\mathrm{conv\_cat}(\mathcal{V}_{nf}, \mathcal{V}_{ns})$), concatenation after convolution ($\mathrm{cat\_conv}(\mathcal{V}_{nf}, \mathcal{V}_{ns}))$), and convolution with our proposed message passing method ($\mathrm{conv}(\mathcal{V}_{nf}, \mathcal{V}_{ns}))$). Incorporating the augmented graph features to $\mathrm{conv}(\mathcal{V}_{nf}, \mathcal{V}_{ns})$ leads to the full CoS-GCN.

The results of our ablation study using the graph classification task are displayed in Table 10. The paired signed-rank test shows when compared with other ablation parts except $\mathrm{conv}(\mathcal{V}_{nf}, \mathcal{V}_{ns})$, the improvement of CoS-GCN across 12 datasets is significant at 99% confidence level. The enhancement of CoS-GCN than $\mathrm{conv}(\mathcal{V}_{nf}, \mathcal{V}_{ns})$ across all datasets is significant at 85% confidence level. In detail, using node features ($\mathcal{V}_{nf}$) or augmented node/graph structural features ($\mathcal{V}_{ns}/\mathcal{V}_{gs}$) solely can achieve good performance, and using $\mathcal{V}_{nf}$ often outperforms $\mathcal{V}_{ns}$ and $\mathcal{V}_{gs}$ on most datasets. This indicates that both the original and augmented features are useful in graph representation learning but the augmented features is limitedly informative. The simple concatenation of $\mathcal{V}_{nf}$ and $\mathcal{V}_{ns}$, *i.e.*, $\mathbf{conv\_cat}(\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}})$ or $\mathbf{cat\_conv}(\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}}))$, helps improve the performance over the using of them solely, indicating the complementary information gained from the graph augmentation relative to the original node features. Our proposed message passing (convolution) method on top of the real and augmented node features, *i.e.*, $\mathbf{conv}(\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}})$, further enhances the results substantially, which demonstrates the effectiveness of our proposed message passing in capturing intricate relations that cannot be captured in the vanilla GCN. Lastly, incorporating the augmented graph-level features would lead to the full model CoS-GNN that largely improves $\mathbf{conv}(\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}})$, demonstrating that the generated global graph structural features are also important for the overall improvement.

Table 9: Results of CoS-GNN with two baselines on three OOD datasets. G-X is short for the dataset name GOOD-X.

G-Motif	**Base**			
	Covariate		Concept	
Accuracy	OOD Validation	ID Validation	OOD Validation	ID Validation
GCN	0.321 ± 0.000	0.343 ± 0.025	0.395 ± 0.014	0.382 ± 0.017
GOOD	0.687 ± 0.034	0.700 ± 0.019	0.814 ± 0.006	0.809 ± 0.007
CoS-GCN	0.868 ± 0.004	0.865 ± 0.003	**0.934 ± 0.000**	**0.932 ± 0.001**
CoS-GIN	**0.888 ± 0.020**	**0.896 ± 0.007**	0.931 ± 0.001	0.923 ± 0.009

G-Motif	**Size**			
	Covariate		Concept	
Accuracy	OOD Validation	ID Validation	OOD Validation	ID Validation
GCN	0.346 ± 0.008	0.350 ± 0.003	0.391 ± 0.0172	0.385 ± 0.018
GOOD	0.517 ± 0.023	0.513 ± 0.019	0.708 ± 0.006	0.694 ± 0.009
CoS-GCN	**0.863 ± 0.043**	**0.816 ± 0.077**	**0.935 ± 0.000**	**0.933 ± 0.002**
CoS-GIN	0.598 ± 0.070	0.555 ± 0.096	0.918 ± 0.006	0.898 ± 0.014

G-HIV	**Scaffold**			
	Covariate		Concept	
AUC	OOD Validation	ID Validation	OOD Validation	ID Validation
GCN	0.669 ± 0.026	0.676 ± 0.016	0.700 ± 0.014	0.607 ± 0.016
GOOD	**0.696 ± 0.020**	0.689 ± 0.021	**0.723 ± 0.010**	**0.653 ± 0.035**
CoS-GCN	0.690 ± 0.017	**0.699 ± 0.023**	0.708 ± 0.009	0.605 ± 0.026
CoS-GIN	0.684 ± 0.021	0.663 ± 0.036	0.722 ± 0.011	0.636 ± 0.016

G-HIV	**Size**			
	Covariate		Concept	
AUC	OOD Validation	ID Validation	OOD Validation	ID Validation
GCN	0.591 ± 0.020	0.580 ± 0.012	0.638 ± 0.0110	0.533 ± 0.009
GOOD	0.600 ± 0.029	0.584 ± 0.025	0.633 ± 0.025	0.448 ± 0.029
CoS-GCN	**0.607 ± 0.019**	**0.619 ± 0.005**	0.654 ± 0.008	0.547 ± 0.007
CoS-GIN	0.585 ± 0.029	0.599 ± 0.028	**0.731 ± 0.006**	**0.622 ± 0.016**

G-SST2	**Length**			
	Covariate		Concept	
Accuracy	OOD Validation	ID Validation	OOD Validation	ID Validation
GCN	0.825 ± 0.008	0.805 ± 0.010	0.724 ± 0.012	0.677 ± 0.010
GOOD	0.813 ± 0.004	0.778 ± 0.011	0.724 ± 0.005	0.673 ± 0.001
CoS-GCN	**0.828 ± 0.010**	**0.814 ± 0.014**	0.730 ± 0.007	**0.685 ± 0.023**
CoS-GIN	0.822 ± 0.012	0.796 ± 0.021	**0.737 ± 0.012**	**0.685 ± 0.013**

Table 10: Results of the ablation study of CoS-GCN in the graph classification task.

Dataset	$\mathcal{V}_{\mathbf{nf}}$	$\mathcal{V}_{\mathbf{ns}}$	$\mathcal{V}_{\mathbf{gs}}$	$\mathbf{conv\_cat}(\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}})$	$\mathbf{cat\_conv}(\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}}))$	$\mathbf{conv}(\mathcal{V}_{\mathbf{nf}}, \mathcal{V}_{\mathbf{ns}})$	**CoS-GCN**
BZR	0.8395 ± 0.0280	$\mathbf{0.8470 \pm 0.0284}$	0.7877 ± 0.0101	0.8075 ± 0.0408	0.8298 ± 0.0248	$\underline{0.8445 \pm 0.0380}$	0.8321 ± 0.0361
COX2	$\underline{0.8113 \pm 0.0432}$	0.7816 ± 0.0081	0.7816 ± 0.0081	0.8050 ± 0.0432	0.8049 ± 0.0444	0.8026 ± 0.0575	$\mathbf{0.8240 \pm 0.0548}$
DD	0.7555 ± 0.0334	0.7436 ± 0.0337	0.7555 ± 0.0375	0.7699 ± 0.0430	$\underline{0.7742 \pm 0.0431}$	0.7725 ± 0.0316	$\mathbf{0.7784 \pm 0.0322}$
I−BINARY	0.7240 ± 0.0364	0.7090 ± 0.0559	0.7060 ± 0.0518	0.7250 ± 0.0686	$\underline{0.7410 \pm 0.0418}$	0.7380 ± 0.0334	$\mathbf{0.7540 \pm 0.0516}$
I−MULTI	0.4980 ± 0.0240	0.4827 ± 0.0320	0.4793 ± 0.0327	0.4947 ± 0.0275	0.4913 ± 0.0257	$\mathbf{0.5087 \pm 0.0237}$	$\underline{0.5027 \pm 0.0189}$
MUTAG	0.7439 ± 0.1021	0.8673 ± 0.0418	0.8076 ± 0.0913	0.8234 ± 0.0886	0.8351 ± 0.0642	$\underline{0.8719 \pm 0.0661}$	$\mathbf{0.8775 \pm 0.0594}$
NCI1	0.7847 ± 0.0161	0.6895 ± 0.0182	0.6343 ± 0.0145	0.7903 ± 0.0168	0.7888 ± 0.0188	$\underline{0.8083 \pm 0.0113}$	$\mathbf{0.8163 \pm 0.0134}$
NCI109	0.7686 ± 0.0263	0.6991 ± 0.0253	0.6293 ± 0.0245	0.7737 ± 0.0177	0.7727 ± 0.0285	$\underline{0.7991 \pm 0.0228}$	$\mathbf{0.8013 \pm 0.0207}$
PROTS_full	0.7493 ± 0.0284	0.7278 ± 0.0306	0.7296 ± 0.0232	$\underline{0.7583 \pm 0.0337}$	0.7521 ± 0.0294	$\mathbf{0.7619 \pm 0.0267}$	0.7574 ± 0.0279
R−BINARY	0.8995 ± 0.0241	$\underline{0.9115 \pm 0.0249}$	0.8295 ± 0.0203	0.9095 ± 0.0268	0.9080 ± 0.0268	0.9110 ± 0.0250	$\mathbf{0.9170 \pm 0.0215}$
R−MULTI	0.5333 ± 0.0129	0.5443 ± 0.0181	0.5031 ± 0.0265	0.5425 ± 0.0149	0.5425 ± 0.0208	$\underline{0.5535 \pm 0.0114}$	$\mathbf{0.5535 \pm 0.0275}$
ENZYMES	0.4500 ± 0.0679	0.2783 ± 0.0325	0.2750 ± 0.0651	0.4317 ± 0.0669	$\underline{0.5017 \pm 0.0626}$	0.4617 ± 0.0738	$\mathbf{0.5333 \pm 0.0641}$
Rank	4.7	4.9	6.6	3.9	3.8	$\underline{2.4}$	**1.5**
p-value	0.0015	0.0034	0.0005	0.0010	0.0005	0.1289	–

Table 11: Efficiency of the augmented node features in graph classification.

	Separated features					Combined features					Completed
	Structural			Quantized		Structural			Quantized	Structural	Completed
Dataset	**w/o Dg**	**w/o Tri**	**w/o CK**	**w/o TCo**	**w/o SCo**	**w/o TCK**	**w/o DT**	**w/o DCK**	**w/o n_quant**	**w/o n_sub**	**CoS-GCN**
BZR	0.847±0.048	0.832±0.043	0.857±0.039	0.835±0.052	0.857±0.048	0.842±0.039	0.845±0.053	0.857±0.042	0.815±0.064	0.832±0.052	0.832±0.036
COX2	0.818±0.049	0.829±0.043	0.814±0.043	0.827±0.045	0.805±0.072	0.807±0.051	0.827±0.062	0.816±0.041	0.803±0.054	0.822±0.056	0.824±0.055
DD	0.774±0.027	0.757±0.028	0.750±0.021	0.753±0.027	0.756±0.032	0.750±0.035	0.743±0.036	0.750±0.024	0.777±0.030	0.770±0.030	0.778±0.032
I-BINARY	0.749±0.040	0.738±0.041	0.726±0.042	0.747±0.026	0.734±0.036	0.731±0.050	0.732±0.041	0.726±0.049	0.750±0.053	0.726±0.034	0.754±0.052
I-MULTI	0.473±0.039	0.498±0.019	0.469±0.031	0.489±0.029	0.486±0.034	0.495±0.024	0.492±0.022	0.458±0.025	0.493±0.033	0.503±0.024	0.503±0.019
MUTAG	0.861±0.093	0.841±0.070	0.803±0.082	0.851±0.066	0.872±0.068	0.808±0.077	0.851±0.057	0.781±0.103	0.862±0.060	0.835±0.077	0.878±0.059
NCI1	0.780±0.019	0.782±0.023	0.782±0.021	0.793±0.016	0.791±0.017	0.775±0.014	0.783±0.022	0.751±0.012	0.816±0.017	0.817±0.013	0.816±0.013
NCI109	0.777±0.025	0.783±0.019	0.767±0.024	0.776±0.018	0.780±0.023	0.760±0.029	0.772±0.025	0.722±0.030	0.796±0.024	0.795±0.022	0.801±0.021
PROTS_full	0.750±0.039	0.749±0.034	0.753±0.039	0.758±0.038	0.757±0.039	0.758±0.042	0.750±0.031	0.748±0.036	0.755±0.025	0.761±0.026	0.757±0.028
R-BINARY	0.915±0.013	0.920±0.009	0.889±0.025	0.915±0.018	0.905±0.018	0.890±0.022	0.918±0.017	0.902±0.025	0.906±0.018	0.895±0.033	0.917±0.022
R-MULTI	0.541±0.015	0.554±0.014	0.523±0.018	0.558±0.021	0.546±0.019	0.528±0.021	0.551±0.013	0.549±0.022	0.557±0.009	0.550±0.022	0.554±0.028
ENZYMES	0.490±0.048	0.568±0.057	0.557±0.074	0.523±0.043	0.545±0.066	0.510±0.051	0.528±0.050	0.503±0.045	0.520±0.059	0.535±0.072	0.533±0.064
Rank	6.3	4.6	7.8	4.8	5.4	7.9	5.9	8.8	5.0	4.9	2.8
p-value	0.0093	0.1387	0.0093	0.0400	0.0986	0.0034	0.0220	0.0049	0.0049	0.0488	–

Table 12: Efficiency of the augmented graph features in graph classification.

	Separated features					Combined features					Completed
	Structural			Quantized		Structural			Quantized	Structural	Completed
Dataset	w/o Tri	w/o Cli	w/o Bri	w/o ClCo	w/o Effi	w/o TBri	w/o ClBri	w/o TrCl	w/o g_quant	w/o g_sub	CoS-GCN
BZR	0.850±0.056	0.852±0.032	0.857±0.043	0.840±0.053	0.840±0.050	0.845±0.037	0.847±0.056	0.845±0.068	0.785±0.023	0.830±0.051	0.832±0.036
COX2	0.812±0.056	0.820±0.052	0.827±0.070	0.812±0.061	0.824±0.048	0.822±0.050	0.799±0.068	0.807±0.055	0.784±0.011	0.798±0.057	0.824±0.055
DD	0.759±0.031	0.769±0.026	0.764±0.029	0.766±0.030	0.764±0.020	0.768±0.034	0.773±0.034	0.761±0.029	0.745±0.044	0.784±0.040	0.778±0.032
I-BINARY	0.744±0.040	0.730±0.035	0.738±0.035	0.733±0.026	0.744±0.042	0.736±0.036	0.731±0.034	0.723±0.036	0.567±0.059	0.736±0.051	0.754±0.052
I-MULTI	0.487±0.028	0.480±0.024	0.475±0.035	0.475±0.024	0.471±0.024	0.477±0.025	0.475±0.028	0.499±0.025	0.366±0.033	0.501±0.026	0.503±0.019
MUTAG	0.830±0.065	0.856±0.067	0.856±0.082	0.867±0.064	0.878±0.058	0.825±0.062	0.856±0.078	0.846±0.060	0.856±0.076	0.861±0.083	0.878±0.059
NCI1	0.796±0.022	0.794±0.013	0.795±0.025	0.753±0.081	0.798±0.017	0.790±0.030	0.787±0.015	0.787±0.027	0.784±0.018	0.810±0.016	0.816±0.013
NCI109	0.783±0.018	0.779±0.016	0.774±0.025	0.772±0.025	0.776±0.028	0.782±0.015	0.781±0.017	0.782±0.024	0.772±0.024	0.799±0.016	0.801±0.021
PROTS_full	0.766±0.046	0.741±0.044	0.750±0.039	0.757±0.042	0.769±0.041	0.762±0.036	0.754±0.044	0.745±0.045	0.742±0.020	0.748±0.029	0.757±0.028
R-BINARY	0.915±0.014	0.915±0.017	0.914±0.017	0.917±0.021	0.920±0.015	0.909±0.029	0.917±0.022	0.917±0.022	0.757±0.045	0.911±0.024	0.917±0.022
R-MULTI	0.551±0.021	0.563±0.024	0.547±0.017	0.543±0.024	0.551±0.019	0.540±0.026	0.560±0.015	0.556±0.019	0.250±0.053	0.552±0.015	0.554±0.028
ENZYMES	0.520±0.049	0.547±0.067	0.537±0.044	0.523±0.085	0.535±0.041	0.552±0.074	0.535±0.068	0.563±0.077	0.173±0.039	0.525±0.040	0.533±0.064
Rank	5.5	5.3	5.5	6.8	4.4	6.0	5.4	5.9	10.3	5.5	2.9
p-value	0.0278	0.0669	0.0542	0.0039	0.168	0.0679	0.0420	0.0977	0.0005	0.0068	–

5.9.2. Ablation Study of the Augmented Features

In this section, we evaluate the effect of each augmented features on the final performance of CoS-GNN. We divided the augmented features into two categories, *i.e.*, one is the characteristics of some specific substructures, and another is some quantized values to measure structural properties of the node/graph. Then we remove each feature and their combinations in each category separately and compare the classification results with our CoS-GNN. The removal of node and graph features are implemented separately. The GNN backbone we use here is CoS-GCN.

Firstly, we delete degree, triangle, clique and k-core (denoted by w/o Dg, w/o Tri, w/o CK respectively) and then remove their combinations, *i.e.*, degree and triangle; degree, clique and k-core; triangle, clique and k-core; all the characteristics (shortened to w/o DT, w/o DCK, w/o TCK, w/o n_sub). We also remove quantized values – triangle clustering coefficient, square clustering coefficient and their combination (written as w/o TCo, w/o SCo and w/o n_quant respectively). The results are reported in Table 11. It is obvious that the removal of augmented features might cause better performance on some specific datasets but will results in decline on many other datasets, leading to a clear decline in the overall performance. Although our CoS-GCN still ranks first on the overall performance, yhe paired signed-rank test indicates that the performance drop of models with part of augmented features across 12 datasets is significant at 85% to 99% confidence level. Deletion of degree and clique and k-core characteristics respectively and their combinations often lead to worse performance, indicating their effect in the full CoS-GCN. Omitting all the node structural characteristics performs better than removing part of them on some datasets and this might be because that the remaining structural characteristics increase the similarity among data.

Later, we delete augmented graph features sequentially (*i.e.*, w/o Tri, w/o Cli, w/o Bri, w/o ClCo and w/o Effi stand for removing triangle, clique, bridge numbers, average clustering coefficient and average local and global efficiency, respectively; w/o TBri, w/o ClBri and w/o TrCl denote deleting triangle and bridge numbers, clique and bridge numbers and triangle and clique numbers; w/o g_quant and w/o g_sub means removing the quantized values and graph substructural statistics, respectively). The results are shown in Table 12. The improvement of our CoS-GCN over the competing methods across the datasets is significant at 80% to 99% confidence level. The removal of triangle, clique, bridge and their combination knowledge results in similar overall performance, which might be because that each feature contributes to the performance of CoS-GNN differently in different dataset. The deletion of average clustering coefficient or all quantized values has larger effect on the final performance, which indicates that the average clustering coefficient information is more discriminative. In summary, the graph-level substructural characteristics are also beneficial in our CoS-GNN since the removal of them leads to a clear decline of the overall performance of CoS-GNN.

6. Conclusion

In this work, we propose a collective structure knowledge-augmented graph neural network (CoS-GNN) to enhance the expressive power of conventional message passing neural networks. The augmented node and graph features carry important and generalizable structural knowledge, which is tapped by our proposed message passing mechanism to integrate the original and augmented graph knowledge, resulting in graph representations with significantly improved expressiveness. This is justified by extensive experiments in various down-stream tasks, including graph classification, anomaly detection, and OOD generalization.

7. Acknowledgments

In this work R. Ma and L. Chen are supported by ARC DP210101347.

References

[1] Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S. V. N., Smola, A. J., & Kriegel, H. P. (2005). Protein function prediction via graph kernels. Bioinformatics, 21(suppl_1), i47-i56. https://doi.org/10.1093/bioinformatics/bti1007.

[2] Borgwardt, K. M., & Kriegel, H. P. (2005, November). Shortest-path kernels on graphs. In Fifth IEEE international conference on data mining (ICDM'05) (pp. 8-pp). IEEE. https://doi.org/10.1109/icdm.2005.132.

[3] Zhao, X., Wu, J., Peng, H., Beheshti, A., Monaghan, J. J., McAlpine, D., ... & He, L. (2022). Deep reinforcement learning guided graph neural networks for brain network analysis. Neural Networks, 154, 56-67. https://doi.org/10.1016/j.neunet.2022.06.035

[4] Song, Z., Yang, X., Xu, Z., & King, I. (2022). Graph-based semi-supervised learning: A comprehensive review. IEEE Transactions on Neural Networks and Learning Systems. https://doi.org/10.1109/TNNLS.2022.3155478.

[5] Wu, Y., Chen, Y., Yin, Z., Ding, W., & King, I. (2023). A survey on graph embedding techniques for biomedical data: Methods and applications. Information Fusion, 100, 101909. https://doi.org/10.1016/j.inffus.2023.101909.

[6] Hassani, K. (2022, June). Cross-domain few-shot graph classification. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 36, No. 6, pp. 6856-6864). https://doi.org/10.1609/aaai.v36i6.20642.

[7] Li, S., Zhou, J., Xu, T., Dou, D., & Xiong, H. (2022, June). Geomgcl: Geometric graph contrastive learning for molecular property prediction. In Proceedings of the AAAI conference on artificial intelligence (Vol. 36, No. 4, pp. 4541-4549). https://doi.org/10.1609/aaai.v36i4.20377.

[8] Zhang, Z., Chen, L., Zhong, F., Wang, D., Jiang, J., Zhang, S., ... & Li, X. (2022). Graph neural network approaches for drug-target interactions. Current Opinion in Structural Biology, 73, 102327. https://doi.org/10.1016/j.sbi.2021.102327.

[9] Zhang, Y., Gao, S., Pei, J., & Huang, H. (2022, August). Improving social network embedding via new second-order continuous graph neural networks. In Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining (pp. 2515-2523). https://doi.org/10.1145/3534678.3539415.

[10] Stanković, L., Mandic, D., Daković, M., Brajović, M., Scalzo, B., Li, S., & Constantinides, A. G. (2020). Data analytics on graphs Part I: Graphs and spectra on graphs. Foundations and Trends® in Machine Learning, 13(1), 1-157. http://dx.doi.org/10.1561/2200000078-1.

[11] Yanardag, P., & Vishwanathan, S. V. N. (2015, August). Deep graph kernels. In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1365-1374). https://doi.org/10.1145/2783258.2783417.

[12] Shen, Y., Jiang, X., Li, Z., Wang, Y., Xu, C., Shen, H., & Cheng, X. (2023). UniSKGRep: A unified representation learning framework of social network and knowledge graph. Neural Networks, 158, 142-153. https://doi.org/10.1016/j.neunet.2022.11.010.

[13] He, H., Ji, Y., & Huang, H. H. (2022, June). Illuminati: Towards explaining graph neural networks for cybersecurity analysis. In 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P) (pp. 74-89). IEEE. https://doi.org/10.1109/eurosp53844.2022.00013.

[14] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems, 32(1), 4-24. https://doi.org/10.1109/tnnls.2020.2978386.

[15] Kipf, T. N., & Welling, M. (2016, November). Semi-Supervised Classification with Graph Convolutional Networks. In International Conference on Learning Representations.

[16] Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in neural information processing systems, 30.

[17] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018, February). Graph Attention Networks. In International Conference on Learning Representations.

[18] Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018, September). How Powerful are Graph Neural Networks?. In International Conference on Learning Representations.

[19] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., & Grohe, M. (2019, July). Weisfeiler and leman go neural: Higher-order graph neural networks. In Proceedings of the AAAI conference on artificial intelligence (Vol. 33, No. 01, pp. 4602-4609). https://doi.org/10.1609/aaai.v33i01.33014602.

[20] Ding, K., Xu, Z., Tong, H., & Liu, H. (2022). Data augmentation for deep graph learning: A survey. ACM SIGKDD Explorations Newsletter, 24(2), 61-77. https://doi.org/10.1145/3575637.3575646.

[21] Li, P., Wang, Y., Wang, H., & Leskovec, J. (2020). Distance encoding: Design provably more powerful neural networks for graph representation learning. Advances in Neural Information Processing Systems, 33, 4465-4478.

[22] Bouritsas, G., Frasca, F., Zafeiriou, S., & Bronstein, M. M. (2022). Improving graph neural network expressivity via subgraph isomorphism counting. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(1), 657-668. https://doi.org/10.1109/tpami.2022.3154319.

[23] You, J., Gomes-Selman, J. M., Ying, R., & Leskovec, J. (2021, May). Identity-aware graph neural networks. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 12, pp. 10737-10745). https://doi.org/10.1609/aaai.v35i12.17283.

[24] Liu, S., Ying, R., Dong, H., Li, L., Xu, T., Rong, Y., ... & Wu, D. (2022, June). Local augmentation for graph neural networks. In International Conference on Machine Learning (pp. 14054-14072). PMLR.

[25] Zhang, M., & Li, P. (2021). Nested graph neural networks. Advances in Neural Information Processing Systems, 34, 15734-15747.

[26] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of big data, 6(1), 1-48. https://doi.org/10.1186/s40537-019-0197-0.

[27] Bayer, M., Kaufhold, M. A., & Reuter, C. (2022). A survey on data augmentation for text classification. ACM Computing Surveys, 55(7), 1-39. https://doi.org/10.1145/3544558.

[28] Sato, R., Yamada, M., & Kashima, H. (2021). Random features strengthen graph neural networks. In Proceedings of the 2021 SIAM international conference on data mining (SDM) (pp. 333-341). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611976700.38.

[29] Liu, X., Cheng, J., Song, Y., & Jiang, X. (2022, June). Boosting graph structure learning with dummy nodes. In International Conference on Machine Learning (pp. 13704-13716). PMLR.

[30] Han, X., Jiang, Z., Liu, N., & Hu, X. (2022, June). G-mixup: Graph data augmentation for graph classification. In International Conference on Machine Learning (pp. 8230-8248). PMLR.

[31] Park, J., Shim, H., & Yang, E. (2022, June). Graph transplant: Node saliency-guided graph mixup with local structure preservation. In Proceed-

ings of the AAAI Conference on Artificial Intelligence (Vol. 36, No. 7, pp. 7966-7974). https://doi.org/10.1609/aaai.v36i7.20767.

[32] Papp, P. A., Martinkus, K., Faber, L., & Wattenhofer, R. (2021). DropGNN: Random dropouts increase the expressiveness of graph neural networks. Advances in Neural Information Processing Systems, 34, 21997-22009.

[33] Maron, H., Ben-Hamu, H., Serviansky, H., & Lipman, Y. (2019). Provably powerful graph networks. Advances in Neural Information Processing Systems, 32.

[34] Grohe, M. (2017). Descriptive complexity, canonisation, and definable graph structure theory (Vol. 47). Cambridge University Press. https://doi.org/10.1017/9781139028868.

[35] Chen, Z., Chen, L., Villar, S., & Bruna, J. (2020). Can graph neural networks count substructures?. Advances in Neural Information Processing Systems, 33, 10383-10395.

[36] Arvind, V., Fuhlbrück, F., Köbler, J., & Verbitsky, O. (2020). On weisfeiler-leman invariance: Subgraph counts and related graph properties. Journal of Computer and System Sciences, 113, 42-59. https://doi.org/10.1016/j.jcss.2020.04.003.

[37] Ju, W., Yang, J., Qu, M., Song, W., Shen, J., & Zhang, M. (2022, February). Kgnn: Harnessing kernel-based networks for semi-supervised graph classification. In Proceedings of the fifteenth ACM international conference on web search and data mining (pp. 421-429). https://doi.org/10.1145/3488560.3498429.

[38] Luo, X., Zhao, Y., Qin, Y., Ju, W., & Zhang, M. (2023). Towards semi-supervised universal graph classification. IEEE Transactions on Knowledge and Data Engineering.

[39] Ju, W., Luo, X., Ma, Z., Yang, J., Deng, M., & Zhang, M. (2022). Ghnn: Graph harmonic neural networks for semi-supervised graph-level classification. Neural Networks, 151, 70-79. https://doi.org/10.1016/j.neunet.2022.03.018.

[40] Ju, W., Gu, Y., Luo, X., Wang, Y., Yuan, H., Zhong, H., & Zhang, M. (2023). Unsupervised graph-level representation learning with hierarchical contrasts. Neural Networks, 158, 359-368. https://doi.org/10.1016/j.neunet.2022.11.019.

[41] Luo, X., Ju, W., Qu, M., Gu, Y., Chen, C., Deng, M., ... & Zhang, M. (2022). Clear: Cluster-enhanced contrast for self-supervised graph representation learning. IEEE Transactions on Neural Networks and Learning Systems. https://doi.org/10.1109/TNNLS.2022.3177775.

[42] Luo, X., Ju, W., Gu, Y., Mao, Z., Liu, L., Yuan, Y., & Zhang, M. (2023). Self-supervised graph-level representation learning with adversarial contrastive learning. ACM Transactions on Knowledge Discovery from Data, 18(2), 1-23. https://doi.org/10.1145/3624018

[43] Chikwendu, I. A., Zhang, X., Agyemang, I. O., Adjei-Mensah, I., Chima, U. C., & Ejiyi, C. J. (2023). A comprehensive survey on deep graph representation learning methods. Journal of Artificial Intelligence Research, 78, 287-356.

[44] Ju, W., Luo, X., Qu, M., Wang, Y., Chen, C., Deng, M., ... & Zhang, M. (2023). TGNN: A joint semi-supervised framework for graph-level classification. arXiv preprint arXiv:2304.11688.

[45] Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., & Neumann, M. (2020). TUDataset: A collection of benchmark datasets for learning with graphs. ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020).

[46] Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., & Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. Journal of Machine Learning Research, 12(9).

[47] Neumann, M., Garnett, R., Bauckhage, C., & Kersting, K. (2016). Propagation kernels: efficient graph kernels from propagated information. Machine learning, 102, 209-245. https://doi.org/10.1007/s10994-015-5517-9.

[48] Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. Advances in Neural Information Processing Systems, 31.

[49] Zhang, Z., Bu, J., Ester, M., Zhang, J., Li, Z., Yao, C., ... & Wang, C. (2021). Hierarchical multi-view graph pooling with structure learning. IEEE Transactions on Knowledge and Data Engineering, 35(1), 545-559. https://doi.org/10.1109/tkde.2021.3090664.

[50] Ma, R., Pang, G., Chen, L., & van den Hengel, A. (2022, February). Deep graph-level anomaly detection by glocal knowledge distillation. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (pp. 704-714). https://doi.org/10.1145/3488560.3498473.

[51] Gui, S., Li, X., Wang, L., & Ji, S. (2022). Good: A graph out-of-distribution benchmark. Advances in Neural Information Processing Systems, 35, 2059-2073.