CLEAR: Cluster-Enhanced Contrast for Self-Supervised Graph Representation Learning

Xiao Luo, Wei Ju, *Graduate Student Member, IEEE*, Meng Qu, Yiyang Gu, Chong Chen, *Member, IEEE*, Minghua Deng, Xian-Sheng Hua, *Fellow, IEEE*, and Ming Zhang

***Abstract*— This article studies self-supervised graph representation learning, which is critical to various tasks, such as protein property prediction. Existing methods typically aggregate representations of each individual node as graph representations, but fail to comprehensively explore local substructures (i.e., motifs and subgraphs), which also play important roles in many graph mining tasks. In this article, we propose a self-supervised graph representation learning framework named cluster-enhanced Contrast (CLEAR) that models the structural semantics of a graph from graph-level and substructure-level granularities, i.e., global semantics and local semantics, respectively. Specifically, we use graph-level augmentation strategies followed by a graph neural network-based encoder to explore global semantics. As for local semantics, we first use graph clustering techniques to partition each whole graph into several subgraphs while preserving as much semantic information as possible. We further employ a self-attention interaction module to aggregate the semantics of all subgraphs into a local-view graph representation. Moreover, we integrate both global semantics and local semantics into a multiview graph contrastive learning framework, enhancing the semantic-discriminative ability of graph representations. Extensive experiments on various real-world benchmarks demonstrate the efficacy of the proposed CLEAR over current graph self-supervised representation learning approaches on both graph classification and transfer learning tasks.**

***Index Terms*— Contrastive learning (CL), graph clustering, graph representation learning, self-supervised learning.**

I. Introduction

GRAPHS have been given increasing prominence as basic representing tools for a variety of structured and sophisticated data, such as social networks [1], [2], road networks [3], and biological protein–protein interaction networks [4], and so on. Typically, a graph is comprised of a finite set of vertices, together with a set of connections implying the relationship between pairs of these vertices (termed edges). Previous researches have investigated numerous aspects of graphs, including node classification [5]–[7], link prediction [8], [9], and graph classification [10], [11]. Among them, a crucial problem therein is to learn discriminative representations of entire graphs, which is vital for a range of domains and tasks, such as predicting protein properties in biological networks [12] and inferring molecule properties in drug discovery [13], [14].

Graph neural networks (GNNs) have achieved remarkable success nowadays in learning graph-level representations [15]–[18]. GNNs generally leverage neighbor-aware message passing mechanisms to produce discriminative node representations. Specifically, each node collects information from its local neighbors, which is then aggregated to iteratively update the node representation. At last, current graph-level representation learning methods embrace graph-level representations by summarizing all the node representations. Typically summarization operations include averaging over all nodes [19] and more complicated graph pooling strategies [16], [17], [20]. In this manner, the learned graph representation can reflect graph structural-semantic information for various downstream applications. Despite their high performance, existing GNN methods are usually trained in a supervised fashion, requiring a large amount of labeled data. However, labeled data is often prohibitively expensive to obtain in many domains [14]. For instance, density functional theory [21] methods are time-consuming to determine the properties of chemical molecule graphs. Meanwhile, in real-world situations, a large number of unlabeled data is always accessible. Therefore, unsupervised or self-supervised graph representation learning approaches are highly anticipated. Traditional graph kernel methods, which resort to handcrafted kernel functions, could result in poor generalization and consequently inferior performance. Inspired by the success in self-supervised contrastive learning (CL), many recent works [22]–[25] have brought the technique to self-supervised graph representation learning. The basic idea of these methods is to augment graphs from multiple perspectives. By enforcing a graph to have similar representations to its own augmented graphs and different representations from other augmented graphs, these methods are able to learn effective graph-level representations, achieving impressive results in many tasks.

Manuscript received 31 December 2021; revised 1 April 2022; accepted 15 May 2022. Date of publication 8 June 2022; date of current version 5 January 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0101902 and in part by the National Natural Science Foundation of China under Grant 12126305 and Grant 31871342. *(Xiao Luo and Wei Ju contributed equally to this work.) (Corresponding authors: Minghua Deng; Ming Zhang.)*

Xiao Luo and Minghua Deng are with the School of Mathematical Sciences, Peking University, Beijing 100871, China (e-mail: xiaoluo@pku.edu.cn; dengmh@pku.edu.cn).

Wei Ju, Yiyang Gu, and Ming Zhang are with the School of Computer Science, Peking University, Beijing 100871, China (e-mail: juwei@pku.edu.cn; yiyanggu@pku.edu.cn; mzhang_cs@pku.edu.cn).

Meng Qu is with Mila—Quebec AI Institute, Université de Montréal, Montreal, QC H3T1J4, Canada (e-mail: meng.qu@umontreal.ca).

Chong Chen and Xian-Sheng Hua are with the Discovery, Adventure, Momentum and Outlook (DAMO) Academy, Alibaba Group, Hangzhou 311100, China (e-mail: cheung.cc@alibaba-inc.com; xiansheng.hxs@alibaba-inc.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3177775.

Digital Object Identifier 10.1109/TNNLS.2022.3177775

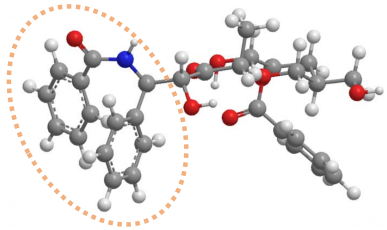

Fig. 1. Chemical example for illustration of the importance of graph substructures. The substructure circled by orange dotted lines contains key multiple rings (i.e., motifs), and thus, indicates the crucial molecule properties.

Typically, existing self-supervised methods [24], [25] compute graph representations by aggregating the representation of every single node, allowing these methods to well capture the node-level semantics. Although nodes are the most important structural attributes in a graph, there are other critical attributes as well, ranging from edges, and motifs to subgraphs. The graph substructures (i.e., motifs and subgraphs) are widely recognized to contain critical characteristics [26]. As depicted in Fig. 1, rings serve as important indicators for property prediction in molecular chemistry [27]. As such, we would also wish to capture the semantic information of local substructures for graph representations, so that the learned representations can capture diverse structural semantics of graphs. However, this problem is nontrivial due to the following challenges: 1) **loss of semantic.** Some methods attempt to sample a subgraph from the whole graph for CL [24], which may lose some vital semantics. For example, in chemistry, the graph property often depends on several dense subgraphs containing key graph motifs. If some of the pivotal patches are not included in the sampled subgraph, the semantics of the graph may be lost, damaging the performance of graph representation learning. 2) **Neglection of intersubstructure information.** The interaction between local substructures usually indicates underlying semantics. To be specific, in biology, some graph motifs may interact and determine the graph property jointly. However, most self-supervised methods [23], [25] are inherent flat, as they only globally summarize node representations by a sum-pooling or mean-pooling [28], failing to model substructure interactions, which is hard to hierarchically embed the accurate semantic information in graphs.

To tackle the above-mentioned challenges, in this article, we propose a principled framework called cluster-enhanced contrast (CLEAR) for self-supervised graph-level representation learning. Our approach models the structural semantics of a graph at graph-level and substructure-level granularities, i.e., global semantics and local semantics, respectively. The global semantics is learned by graph-level augmentation strategies followed by a GNN-based encoder, which learns the node representations and aggregates them into a global-view graph representation. To capture the local semantics, we partition the whole graph into several clusters (i.e., subgraphs) by utilizing graph clustering algorithms [29], [30]. These algorithms are able to damage as few as possible intersubgraph edges. Thus the pivotal patches are typically contained in one or several subgraphs. In this way, we can sufficiently explore underlying semantics from the local view. After encoding each subgraph via the GNN-based encoder, we further leverage a self-attention interaction module to first model subgraph interactions and then effectively aggregate different subgraph representations as well as subgraph interaction representations into a local-view graph representation. To combine both global- and local-view representations, we propose a multiview CL framework that guides the network to learn consistent representations at different granularities, enhancing the semantic-discriminative ability of graph-level representations. We validate our proposed CLEAR on a wide range of graph classification benchmark datasets and large-scale OGB datasets. The results demonstrate that our CLEAR achieves state-of-the-art performance by significantly outperforming the baselines on both graph classification and transfer learning tasks. The main contributions in this article can be highlighted as follows.

1) To explore the semantics of local substructures, we construct local-view graph representations while mining as much semantic information as possible by random graph clustering and subgraph-interaction modeling.
2) We introduce a unified self-supervised graph representation learning framework CLEAR. Benefiting from CL at different granularities, CLEAR can well produce discriminative graph representations.
3) Extensive experiments have shown that the proposed method outperforms current state-of-the-art methods on a wide range of datasets in both graph classification and transfer learning downstream tasks.

II. Related Work

In this section, we introduce some related fields, problems, and works involved in our research. We begin by providing a brief review of graph representation learning. Then, we introduce the existing graph CL approaches. Finally, we introduce several graph clustering techniques.

A. Graph Representation Learning

In recent years, the emphasis on graph mining has evolved away from structural feature engineering and toward graph representation learning [31], which includes node-level [32] and graph-level representation learning [22]. While the former has been intensively investigated recently with considerable success, the latter is understudied yet critical to the performance of graph machine learning models in abundant domains, such as drug discovery and bioinformatics. With the development of neural networks, GNNs [5], [28], [33], [34] have been widely utilized for learning useful graph-level representations in many fields. The majority of existing GNN methods inherently follow a message-passing scheme [35] recursively to embed graphs into a continuous and low-dimensional space. Despite their effectiveness in learning graph-level representations, these approaches are usually trained in a supervised fashion, which requires a large amount of labeled data for training. Unfortunately, annotating samples is often costly, hindering these methods from real-world practice [14]. To tackle this issue, our work studies self-supervised graph representation learning, which fully explores the semantic information of the unlabeled data from different granularities.

B. Graph Contrastive Learning

Recently, CL utilizes self-supervised information between contrastive pairs generated via stochastic perturbation of the initial inputs, achieving remarkable success in various fields, e.g., computer vision [36], [37] and natural language processing. Extensive efforts have been made to incorporate CL into GNNs [24], [25], [38] recently. As pioneering efforts, Deep Graph Infomax (DGI) [38] and InfoGraph [22] maximize the mutual information between local and global representations on the augmented graphs from node-level and graph-level, respectively. Subsequent techniques [23]–[25] are often built on the framework of visual CL, which enforces a graph to have similar representations to its own augmented graphs and different representations from other augmented graphs. CuCo [23] borrows the idea of curriculum learning to sort negative samples for graph CL. JOint Augmentation Optimization (JOAO) [25] proposes to automatically select dynamical and adaptive graph augmentation strategies for various datasets. Graph contrastive learning with adaptive augmentation (GCA) [32] develops adaptive graph augmentations to incorporate various priors of the graphs. However, these methods generally suffer from failing to comprehensively capture semantic information in local substructures, while our work better explores the semantic information via graph clustering and explicitly incorporates the intersubstructure information.

C. Graph Clustering

Another category of related work is graph clustering [39]–[41], which aims to identify disjoint partitions of graph nodes where connections between nodes within the same partition are significantly denser than connections between nodes across partitions. There have been several graph clustering algorithms for this task. Spectral methods that minimize weighted cuts [42] form an important class of such algorithms and are shown to be effective for problems, such as image segmentation [43]. Multilevel graph partitioning algorithms, such as Metis [44] are well known to scale well for some large-scale graph datasets. Graclus [30] optimizes weighted cuts or normalized cuts by optimizing an equivalent weighted kernel K-means objective. Recently, graph clustering has been applied to different tasks in various domains. For example, Cluster-GCN [45] utilizes graph clustering to improve the memory and computational efficiency of GNNs for node classification. GCAGC [46] proposes to use the graph to discriminate the common objects, boosting the performance of co-saliency detection. In contrast to these supervised tasks, our work utilizes graph clustering algorithms in unsupervised scenarios, which efficiently capture semantic information from a local view. To the best of our knowledge, we are the first to integrate graph clustering into the self-supervised graph representation learning task.

III. Problem Definition

In this section, we first briefly present formal terminology and then introduce the problem formulation.

Definition 1 (Graph): Let $G = (V, E, \mathbf{X})$ represent a graph with V representing the vertex set and E representing the edge set. x_v denotes the feature vector of the node v and $\mathbf{X} \in \mathbb{R}^{|V| \times d^f}$ denotes the node feature matrix, where d^f is the dimension of features.

Self-supervised graph representation learning is a fundamental topic in data mining with a wide range of applications, including predicting protein properties and inferring chemical compound functionalities. It is beneficial to train a graph encoder that can generate informative representations of an entire graph from a collection of unlabeled graphs. Formally, the self-supervised graph representation learning problem is defined as follows.

Definition 2 (Self-Supervised Graph Representation Learning): Given a set of unlabeled graphs $\mathcal{G} = \{G_1, \ldots, G_N\}$, we aim to learn an embedding vector $\mathbf{z}_i \in \mathbb{R}^d$ for each graph G_n under the guidance of the data itself, where d is dimension of hidden embeddings and N is the number of graphs. The generated embeddings $\{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ will be used for downstream applications, such as graph property prediction.

IV. Proposed Method

In this part, we introduce our proposed framework CLEAR.

A. Framework Overview

This article presents the CLEAR for self-supervised graph representation learning, which models the structural semantics of a graph at graph-level and substructure-level granularities, i.e., global semantics and local semantics, respectively. It has been recognized that local substructures in graphs can carry critical characteristics. However, capturing their semantic information of them for representation learning is challenging due to the potential loss of semantic and neglection of intersubstructure information. As a consequence, prior works fail to achieve satisfactory performance when label annotations are very scarce in real-world applications.

In order to better explore local substructures in graphs, the local semantics is learned using graph clustering as well as a self-attention interaction module to aggregate both subgraph representations and subgraph interaction representations. Moreover, the global semantics is learned via graph-level augmentation followed by a GNN-based encoder. Finally, we integrate both global semantics and local semantics into a unified multiview CL framework. More details are illustrated in Fig. 2.

The remainder of this section is organized as follows. To begin, we briefly introduce our GNN-based encoder in Section IV-B. Then, we introduce graph augmentation strategies to generate global-view graph representations in Section IV-C. Moreover, in Section IV-D, we demonstrate the manner to explore local semantics to generate local-view graph representations in detail. Finally, we formulate a multiview CL framework in Section IV-E and analyze the computational complexity in IV-F.

B. GNN-Based Encoder

GNNs have currently occurred as effective approaches for learning graph-structured data. Existing techniques for acquiring graph-level representations are on the basis of message passing neural networks [35]. Particularly, for each node $v \in V$, the embeddings from its neighbors at layer $k-1$ are first

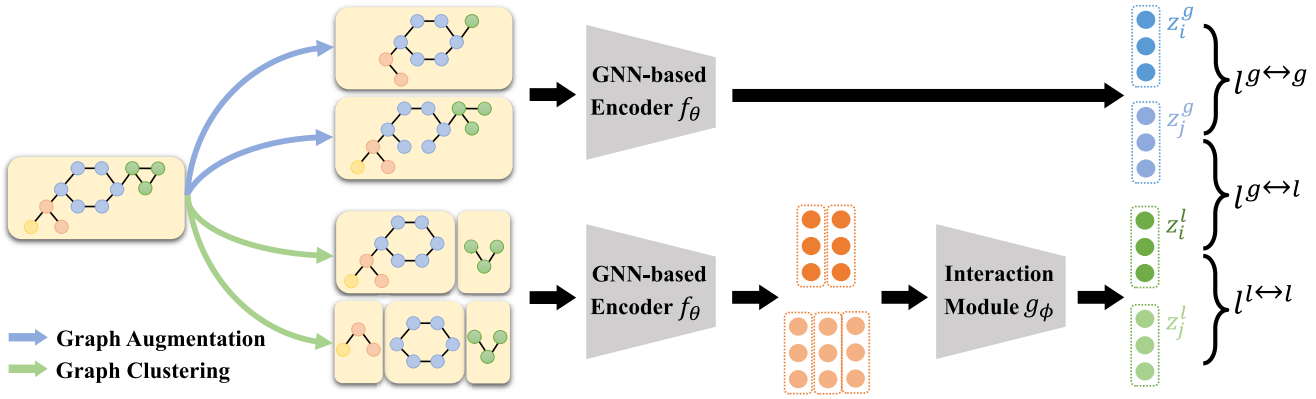

Fig. 2. Illustration of the proposed framework CLEAR. Our method first uses graph augmentation to generate global-view representations while using graph clustering algorithms and a self-attention interaction module to generate local-view representations. Finally, the multiview CL framework is built across global and local views.

aggregated. Then, the representation of node v at layer k, namely, $\mathbf{h}_v^{(k)}$ is updated recursively by merging its embedding in the previous layer with its neighbor embedding. In formulation, $\mathbf{h}_v^{(k)}$ is calculated as follows:

$$\mathbf{h}_v^{(k)} = \mathcal{M}_\theta^{(k)}\left(\mathbf{h}_v^{(k-1)}, \mathcal{A}_\theta^{(k)}\left(\left\{\mathbf{h}_u^{(k-1)}\right\}_{u\in\mathcal{N}(v)}\right)\right) \tag{1}$$

in which $\mathcal{N}(v)$ represents the neighborhood of v. $\mathcal{A}_\theta^{(k)}$ and $\mathcal{M}_\theta^{(k)}$ represent the aggregation and combination operations at layer k, respectively. At last, a readout function is adopted to summarize all the obtained node representation at the Kth layer into the graph-level representation. Formally

$$f_\theta(G) = \text{READOUT}\left(\left\{\mathbf{h}_v^{(K)}\right\}_{v\in V}\right) \tag{2}$$

where $f_\theta(G)$ is the graph-level representation and θ is the parameter set of the encoder.

C. Global-View Graph Representation

Data augmentation is critical in self-supervised learning which produces novel rational data through applying certain transformations without influencing the semantics [36]. Here, we focus on graph-level augmentations, which randomly disturb some nodes, edges, and attributes in a whole graph while maintaining most graph-level information (i.e., global semantics). Specifically, given a graph G, we generate the augmented graph $\hat{G}^g$ satisfying: $\hat{G}^g \sim \mathcal{T}(\hat{G}^g|G)$, in which $\mathcal{T}(\cdot|G)$ is the predefined augmentation distribution conditioned on the initial graph, incorporating the prior of data distribution. We use four popular strategies to augment topological and attributive information of the graph [25], as shown in Fig. 3.

1) *Edge Deletion:* We remove partial edges from each graph obeying an i.i.d. uniform distribution. The dropping probability of each edge follows an i.i.d. uniform distribution by default. The underlying prior is that missing certain edges does not influence the semantics of the graphs in most cases.
2) *Node Deletion:* We randomly select certain nodes and remove them as well as their related edges from the graph. We also follow a uniform distribution to delete each edge. The underlying prior is that the semantic information of graphs is mostly stable when parts of nodes are missing.
3) *Attribute Masking:* We pick partial nodes and mask part of their attributes at random. The strategy assumes that missing partial node attributes will not hugely change the semantic information.
4) *Graph Diffusion:* We randomly transform a graph using diffusion [47] to provide a congruent perspective. This augmentation strategy contributes to providing additional global information. Given the transition matrix T which is used to transfer the adjacency matrix, the diffusion matrix S can be formulated as

$$S = \sum_{w=0}^{\infty} \theta_r T^w \tag{3}$$

where θ is a coefficient to balance the distribution of local and global signals. We use the Personalize PageRank (PPR) kernel to characterize graph diffusion in this article. Specifically, given the adjacency matrix A,

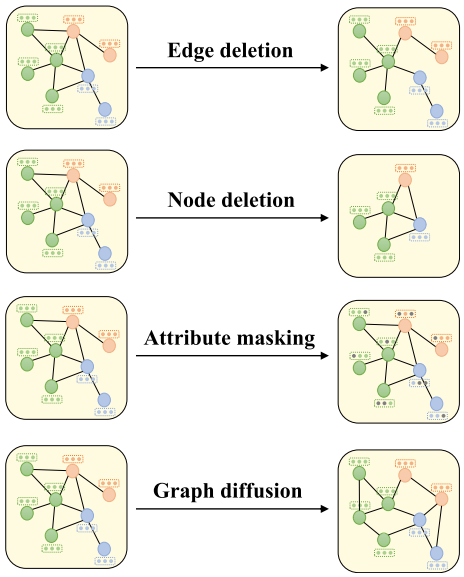

Fig. 3. Illustration of four types of adopted graph augmentation strategies. We randomly select one of four augmentation strategies to produce global-view graph representations.

the degree matrix D and the identity matrix I, the diffusion matrix S is reformulated as

$$S = \alpha\left(I - (1-\alpha)D^{-1/2}AD^{-1/2}\right)^{-1} \tag{4}$$

where α is a random coefficient sampled in $(0, 1)$ as the random walk teleport probability [48]. The produced diffusion matrix S will guide the message passing process in the GNN-based encoder.

The default augmentation ratio for edge/node deletion and masking is set to 0.2 empirically following recent researches [24], [25]. For example, about 20% edges are deleted from the graph using the strategy of the edge deletion. Here, we first generate the augmentations by randomly selecting one of the four augmentation strategies and then use the GNN-based encoder to extract novel rational global-view graph representations as follows:

$$\mathbf{z}^g = f_\theta(\hat{G}^g). \tag{5}$$

D. Local-View Graph Representation

As mentioned in the introduction, local substructures in a graph are vital for learning graph-level representations. However, effectively exploiting semantics in substructures (i.e., local semantics) is challenging. For example, existing subgraph augmentation strategies [24] may lose some underlying semantics. As such, we need to design a module to explore local substructure patterns without loss of underlying semantics and then encourage the substructures to be discriminative for graph representation learning.

1) Random Clustering Strategy: Graph clustering algorithms [30] target generating the partitions over nodes in the graph such that within-cluster links greatly outnumber between-cluster links, which allows for better exploration of the clustering and community structure in the graph. These are exactly what we need because we decompose the original graph into several subgraphs and the loss of semantics is usually proportional to between-cluster links [45]. Besides, the semantics of the graph is usually embedded in the dense patch of the graph, implying that graph clustering can mostly preserve underlying semantics in the original graph. From this point, we propose a unified pipeline to explore underlying semantics from a local view, which first uses graph clustering algorithms to partition the graph and then aggregate the local semantics with a self-attention interaction module. In detail, given a set of cluster numbers $\mathcal{C}$, we first randomly choose a number $C \in \mathcal{C}$, and then partition nodes in the graph into C groups through efficient Metis [44]: $V = [V_1, \ldots, V_C]$, where V_t consists of the nodes in the tth partition. Thus, we have C subgraphs as follows:

$$[G^{l_1}, \ldots, G^{l_C}] = \left[\{V^1, E^1\}, \ldots, \{V^C, E^C\}\right] \tag{6}$$

where each E^c only consists of the links between nodes in V^c. We feed these subgraphs to the GNN-based encoder and get subgraph representations $\mathcal{Z}^l = \{\mathbf{z}^{l_1}, \ldots, \mathbf{z}^{l_C}\}$.

Remark 1: In our graph clustering strategy, the cluster number is randomly chosen in $\mathcal{C}$, which is not a preset parameter. This design extends the fixed graph clustering operation to a random augmentation strategy, which has the following strengths for the exploration of local semantic information. First, our random strategy has the potential to explore varying local substructures, which fits the condition that different graphs could have local semantic information of different sizes. Second, a fixed clustering strategy may lead to the permanent removal of crucial edges, thus bringing in biased semantic information and hindering efficient representations while our random strategy can release this issue. Third, the random clustering strategy tends to emphasize and encourage stable and reliable substructures that are always inseparable in different partitions. Intuitively, these substructures are significant indicators of graph semantics, which can promote effective graph representation learning.

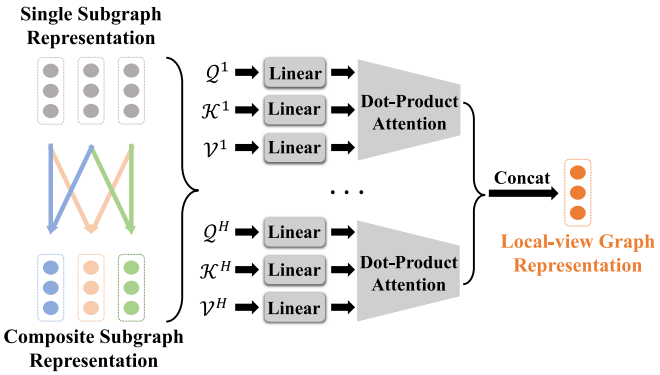

Fig. 4. Illustration of the self-attention interaction module $g_\phi(\cdot)$. We take three clusters as an example. The composite subgraph representations are first generated by mutual vector element-wise product of single-subgraph representations. Then, all representations are aggregated by a multihead attention network, producing a local-view graph representation.

Remark 2: In our model, Metis is adopted following [45] due to its efficiency and scalability in the PyTorch Geometric library [49]. We have tested other graph clustering methods, such as Graclus [30] and weighted cuts [50] with random cluster sizes, which do not bring much change to the performance.

2) Self-Attention Interaction Module: Of note, graph motifs may interact and determine the graph property jointly. Inspired by attention mechanism [51] and factorization machines [52], we use a self-attention interaction module $g_\phi(\cdot)$ to aggregate the subgraph representations into a local-view representation $\mathbf{z}^l = g_\phi(\mathcal{Z}^l)$ in Fig. 4, where ϕ represents the module parameter set. Specifically, we first produce composite subgraph representations to model the subgraph interaction information, which resulting in $C(C-1)/2$ composite embedding vectors: $\mathcal{I}^l = \{\mathbf{z}^{l_{1,2}}, \mathbf{z}^{l_{1,3}}, \ldots, \mathbf{z}^{l_{c-1,c}}\}$, where $\mathbf{z}^{l_{i,j}} = \mathbf{z}^{l_i} \odot \mathbf{z}^{l_j}$ and $\odot$ denotes the elementwise product of vectors. We unite $\mathcal{Z}^l$ and $\mathcal{I}^l$ to obtain a hybrid feature set $\hat{\mathcal{Z}}^l$ involved in $C(C+1)/2$ embedding vectors, which are re-ordered as $\{\hat{\mathbf{z}}^{l_1}, \ldots, \hat{\mathbf{z}}^{l_{C(C+1)/2}}\}$. We then use a multihead self-attention network to aggregate these embedding vectors. Each embedding $\hat{\mathbf{z}}^{l_r}$ is mapped into query vector and key vector, and we calculate their dot product as the attention to learn the importance of semantics underlying the feature vector. Formally, we define $\mathcal{Q}^h, \mathcal{K}^h$ and $\mathcal{V}^h \in \mathbb{R}^{(d/H)\times d}$ as transformation matrices of a query, a key and a value, respectively, where H is the number of heads. Then, the weight α_r^h is determined by the dot-product

of the query and the key as follows:

$$\alpha_r^h = \frac{\left(\mathcal{Q}^h \cdot \mathbf{z}_i^{l_r}\right)^\top \left(\mathcal{K}^h \cdot \mathbf{z}_i^{l_r}\right)}{\sqrt{d/H}}, \quad \hat{\alpha}_r^h = \frac{\exp \alpha_r^h}{\sum_{r'=1}^{C(C+1)/2} \exp \alpha_{r'}^h} \tag{7}$$

where $\hat{\alpha}_r^h$ is viewed as the importance of the rth embedding. Finally, we aggregate subgraph representations and subgraph interaction representations to obtain a local-view graph representation $\mathbf{z}^l$ with the multihead operation

$$\mathbf{z}^l = \Big\Vert_{h=1}^{H} \sum_{r=1}^{C(C+1)/2} \hat{\alpha}_r^h \mathcal{V}^h \cdot \hat{\mathbf{z}}^{l_r} \tag{8}$$

where $\|$ is the concatenation operator.

Next, we will integrate our local- and global-view representations into a unified CL framework.

E. Multiview Contrastive Learning Framework

Our CLEAR is a concise and effective CL framework to enhance both global-view and local-view graph representations. From the global view, a graph G performs stochastic graph augmentations $\mathcal{T}(\cdot|G)$ to obtain two correlated views $\hat{G}_i^g$ and $\hat{G}_j^g$, as a positive pair. Then, graph-level representations $\mathbf{z}_i^g$ and $\mathbf{z}_j^g$ are extracted by the GNN-based encoder $f_\theta(\cdot)$ for $\hat{G}_i^g$ and $\hat{G}_j^g$ respectively. The noise-contrastive estimation loss [53] is utilized to maximize the consistency between positive pairs $\{\mathbf{z}_i^g, \mathbf{z}_j^g\}$ compared with negative pairs. Specifically, a minibatch of M graphs are randomly sampling, producing $2M$ random augmented graphs $\{\hat{G}_{m,i}, \hat{G}_{m,j}\}_{m=1}^M$. For each positive pair $\hat{G}_{m,i}$ and $\hat{G}_{m,j}$, we consider the other $(M-1)$ augmented graphs within a minibatch as negative samples following [24] and [25]. For convenience, we re-annotate $\mathbf{z}_i^g$ and $\mathbf{z}_j^g$ as $\mathbf{z}_{m,i}^g$ and $\mathbf{z}_{m,j}^g$ for the mth graph in the minibatch. Then, two graph representations for the mth graph are compared as follows:

$$\ell_m^{g\leftrightarrow g} = -\log \frac{e^{\mathbf{z}_{m,i}^g \star \mathbf{z}_{m,j}^g/\tau}}{\sum_{m'=1}^M e^{\mathbf{z}_{m,i}^g \star \mathbf{z}_{m',j}^g/\tau}} \tag{9}$$

where $\mathbf{z}_{m,i}^g \star \mathbf{z}_{m,j}^g$ denotes the cosine similarity of $\mathbf{z}_{m,i}^g$ and $\mathbf{z}_{m,j}^g$ and τ is a temperature parameter set to 0.5 following [24], [25], and [36].

However, global-view CL cannot fully capture the underlying semantics within local substructures. To tackle this issue, we develop both global and local CL to keep consistent graph representations on both the whole graph and local subgraph set. This strategy aims to take advantage of global graph representations to enhance local subgraph representations and vice versa. In detail, given a graph G, first we randomly select two cluster numbers C and C' in set $\mathcal{C}$. After graph clustering, we obtain two subgraph sets $\mathcal{Z}_i^l = \{G_i^{l_1}, \ldots, G_i^{l_C}\}$ and $\mathcal{Z}_j^l = \{G_j^{l_1}, \ldots, G_j^{l'_C}\}$. We feed these subgraphs to the GNN-based encoder and self-attention interaction module to get two local-view graph-level representations $\mathbf{z}_i^l$ and $\mathbf{z}_j^l$ from $\mathcal{Z}_i^l$ and $\mathcal{Z}_j^l$, respectively. Similarly, we re-annotate $\mathbf{z}_i^l$ and $\mathbf{z}_j^l$ as $\mathbf{z}_{m,i}^l$ and $\mathbf{z}_{m,j}^l$ for the mth graph in the minibatch. In our framework, we not only contrast two local-view representations but also contrast a local-view representation and a global-view representation. The former aims to generate a consistent representation for substructures of different cluster scales, while the latter aims to produce consistent graph representations for both local substructures and the whole graph. Formally, the local-view CL loss for the mth graph is formulated as

$$\ell_m^{l\leftrightarrow l} = -\log \frac{e^{\mathbf{z}_{m,i}^l \star \mathbf{z}_{m,j}^l/\tau}}{\sum_{m'=1}^M e^{\mathbf{z}_{m,i}^l \star \mathbf{z}_{m',j}^l/\tau}}. \tag{10}$$

Besides, the global-and-local CL loss is

$$\ell_m^{g\leftrightarrow l} = -\log \frac{e^{\mathbf{z}_{m,j}^g \star \mathbf{z}_{m,i}^l/\tau}}{\sum_{m'=1}^M e^{\mathbf{z}_{m,j}^g \star \mathbf{z}_{m',i}^l/\tau}}. \tag{11}$$

In summary, for the mth graph we produce three positive pairs $(\mathbf{z}_{m,i}^g, \mathbf{z}_{m,j}^g)$, $(\mathbf{z}_{m,j}^g, \mathbf{z}_{m,i}^l)$, and $(\mathbf{z}_{m,i}^l, \mathbf{z}_{m,j}^l)$ for CL. The overall loss function of CLEAR for the mth graph in a minibatch can be defined as follows:

$$\ell_m = \frac{1}{3}\left(\ell_m^{g\leftrightarrow g} + \ell_m^{g\leftrightarrow l} + \ell_m^{l\leftrightarrow l}\right). \tag{12}$$

The final loss is computed across all positive pairs in the minibatch. The parameters in our framework are optimized with the minibatch stochastic gradient descent (SGD) algorithm effectively. The whole learning procedure of our proposed model CLEAR is shown in Algorithm 1. The final graph embedding of each graph is produced by our trained encoder for various downstream tasks.

Algorithm 1 Learning Algorithm of CLEAR

Input: Unlabeled data $\mathcal{G}$, batch size M, embedding dimension d, propagation layer number L

```
Initialize model parameter with a Xavier initialization.
while not convergence do
  Sample M samples from G and construct a minibatch.
  for the m-th graph in the minibatch do
    Generate two graph views Ĝ^g_{m,i} and Ĝ^g_{m,j} via graph augmentation.
    Obtain global-view graph representations z^g_{m,i} and z^g_{m,j} through f_θ(·).
    Sample two different cluster number C and C' and cluster the origin graph into subgraph sets.
    Obtain local-view graph representations z^l_{m,i} and z^l_{m,j} through f_θ(·) and g_φ(·).
  end for
  Optimize model parameters θ and φ with Eq.12.
end while
return GNN encoder f_θ(·) and interaction module g_φ(·).
```

F. Complexity Analysis

The computing complexity of Algorithm 1 mainly depends on graph clustering and neural network computation. We will show that the complexity of graph clustering always takes a small percentage of the whole computational time (less than 21%) in Section V-F thanks to our efficient clustering algorithm Metis [44].

TABLE I
STATISTICS OF THE DATASETS

Datasets	Category	Statistics				Labels / Attributes			
		Graphs	Classes	Nodes (avg.)	Edges (avg.)	Node Labels	Edge Labels	Node Attr.	Edge Attr.
PROTEINS [54]	Bioinformatics	1113	2	39.06	72.82	✓	✗	✓	✗
DD [55])	Bioinformatics	1178	2	284.32	715.66	✓	✗	✗	✗
IMDB-B [56]	Social Networks	1000	2	19.77	96.53	✗	✗	✗	✗
IMDB-M [56]	Social Networks	1500	3	13.00	65.94	✗	✗	✗	✗
REDDIT-B [56]	Social Networks	2000	2	429.63	497.75	✗	✗	✗	✗
REDDIT-M-5k [56]	Social Networks	4999	5	508.52	594.87	✗	✗	✗	✗
COLLAB [56])	Social Networks	5000	3	74.49	2457.78	✗	✗	✗	✗

Therefore, our computing complexity mainly depends on the encoder and the interaction modules Step 6 and Step 8. For each graph G, $||A||_0$ represents the number of nonzeros in its adjacency matrix. d^f represents the feature dimension. K represents the GNN layer number. $|V|$ represents the total number of nodes. The time complexity of obtaining a global representation and all subgraph representations is both $\mathcal{O}(K||A||_0 d^f + K|V|{d^f}^2)$ for each graph, while the time complexity of the interaction module is negligible when cluster number C is limited in our application. As a result, the complexity of our CLEAR and the representative graph clustering methods (GraphCL [24], JOAO [25], and CuCo [23]) are all $\mathcal{O}(K||A||_0 d^f + K|V|{d^f}^2)$ for each graph, linearly related to both $|V|$ and $||A||_0$.

V. EXPERIMENTS

In this section, we evaluate the effectiveness of the proposed CLEAR in both graph classification and transfer learning tasks and compare it with a collection of state-of-the-art graph representation learning approaches.

A. Experimental Settings

1) Evaluation Dataset: We perform extensive experiments on several popular graph datasets[1] following [22] and [24], including two bioinformatics and five social network datasets. The bioinformatics datasets include PROTEINS [54] and P. D. Dobson and A. J. Doig (DD) [55], while the social network datasets contains Internet Movie DataBase (IMDB)-B, IMDB-M, REDDIT-B, REDDIT-M-5k, and COLLAB [56]. Following recent works [22], we adopt all-ones vectors as input node attributes if they are not available in these datasets. The detailed statistics are summarized in Table I.

2) Compared Methods: To evaluate the effectiveness of our models, we select three families of baselines, including graph kernel methods, traditional graph embedding methods, and graph CL methods.

The graph kernel methods include: 1) **Graphlet Kernel** [57]—it introduces graphlets that is a kernel based on counting occurrences of substructures of a small-size size for graph pair comparison. 2) **Shortest Path (SP) Kernel** [58]—the basic idea of the SP kernel is to measure the similarity of the attributes and lengths of the SPs connecting all pairs of nodes in two graphs. 3) **Weisfeiler–Lehman (WL) Kernel** [59]—It is inspired by the Weisfeiler–Lehman test [62] of isomorphism on graphs and follows the paradigm of the iterative relabeling process. 4) **Deep Graph Kernel (DGK)** [56]—It learns the latent representations of subgraph patterns via leveraging the dependency information between substructures inspired by language modeling and deep learning.

The traditional graph embedding methods include: 1) **Sub2Vec** [60]—it formulates a subgraph embedding problem and presents an unsupervised approach to learning feature representations of arbitrary subgraphs. 2) **Graph2Vec** [61]—it is a neural embedding framework to learn data-driven distributed representations of graphs using neural networks in an unsupervised manner.

The graph CL methods include in the following: 1) **InfoGraph** [22]—it maximizes the mutual information between the graph-level representation and patch-level representations at different granularity (e.g., nodes, edges, and triangles) to learn whole-graph representations. 2) **GraphCL** [24]—it designs four types of graph augmentations to incorporate various priors, including node dropping, edge perturbation, attribute masking, and subgraph following the scheme in visual CL [36]. 3) **CuCo** [23]—it leverages the idea of curriculum learning and studies the impact of negative samples on learning graph-level representations following the framework of CL. 4) **JOAO** [25]—it presents a bilevel optimization model to choose appropriate data augmentation strategies for specific datasets in an automatic and adaptive manner.

3) Parameter Settings: We implement our CLEAR and baseline methods in Scikit-learn and PyTorch. We use Graph Isomorphism Network (GIN) [28] to parameterize the GNN-based encoder in our CLEAR due to its effectiveness, which consists of three graph convolutional layers followed by a sum-pooling layer. To provide rigorous comparative analysis, we also use GIN when comparing graph CL methods. The cluster number set $\mathcal{C}$ is set to $\{2, 3, 4, 5, 6\}$. We have tested $\max\{C \in \mathcal{C}\}$ from 3 to 10 by validation, and find that $\max\{C \in \mathcal{C}\} = 6$ is enough for all datasets. A larger maximum does not bring much improvement in performance. For a fair comparison, we set the batch size to 64 and the number of epochs to 20 for all the compared methods. For all datasets, the embedding dimensions of the hidden layers are 64. We set the number of attention heads as 2 for simpleness following [63]. The model is optimized with an Adam optimizer [64] with the initial learning rate set to 0.01 and weight decay set to 0.0005. The parameters for all baseline approaches are carefully tuned to obtain the best performance. The source

[1]https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets

TABLE II

RESULTS OF GRAPH CLASSIFICATION ACCURACY (IN %) ON DIFFERENT BENCHMARK DATASETS. THE REPORTED RESULTS ARE MEAN AS WELL AS THE STANDARD DEVIATION OF PREDICTION ACCURACY OVER FIVE RUNS USING DIFFERENT RANDOM SEEDS. BOLD RESULTS INDICATE THE BEST PERFORMANCE

Methods	Reference	Datasets						
		PROTEINS	**DD**	**IMDB-B**	**IMDB-M**	**REDDIT-B**	**REDDIT-M-5k**	**COLLAB**
Graphlet [57]	AISTATS09	71.67 ± 0.55	72.54 ± 3.83	65.87 ± 0.98	43.89 ± 0.38	77.34 ± 0.18	41.01 ± 0.17	56.30 ± 0.60
SP [58]	ICDM05	75.07 ± 0.54	75.47 ± 3.46	55.60 ± 0.22	37.99 ± 0.30	64.11 ± 0.14	39.55 ± 0.22	49.80 ± 1.20
WL [59]	JMLR11	72.92 ± 0.56	74.02 ± 2.28	72.30 ± 3.44	46.95 ± 0.46	68.82 ± 0.41	46.06 ± 0.21	69.30 ± 3.44
DGK [56]	KDD15	73.30 ± 0.82	74.85 ± 0.74	66.96 ± 0.56	44.55 ± 0.52	78.04 ± 0.39	41.27 ± 0.18	64.66 ± 0.50
Sub2Vec [60]	PAKDD18	53.03 ± 5.55	54.33 ± 2.44	55.26 ± 1.54	36.67 ± 0.83	71.48 ± 0.41	36.68 ± 0.42	55.26 ± 1.54
Graph2Vec [61]	arXiv17	73.30 ± 2.05	70.32 ± 2.32	71.10 ± 0.54	46.32 ± 1.44	75.78 ± 1.03	47.86 ± 0.26	71.10 ± 0.54
InfoGraph [22]	ICLR20	74.44 ± 0.53	72.85 ± 1.78	71.11 ± 0.88	49.69 ± 0.53	82.50 ± 1.42	53.46 ± 1.03	70.65 ± 1.13
GraphCL [24]	NeurIPS20	74.39 ± 0.45	78.62 ± 0.40	71.14 ± 0.44	48.49 ± 0.63	84.69 ± 1.40	55.99 ± 0.28	71.36 ± 1.15
CuCo [23]	IJCAI21	74.32 ± 0.31	76.93 ± 0.83	70.47 ± 0.31	47.97 ± 0.12	85.83 ± 1.19	55.78 ± 0.21	69.43 ± 0.26
JOAO [25]	ICML21	74.55 ± 0.41	77.32 ± 0.54	70.21 ± 3.08	47.22 ± 0.41	85.29 ± 1.35	55.74 ± 0.63	69.50 ± 0.36
CLEAR	Ours	**76.10 ± 0.35**	**79.37 ± 0.32**	**72.51 ± 0.81**	**50.26 ± 0.23**	**89.32 ± 1.27**	**56.73 ± 0.69**	**71.65 ± 0.33**

code of CLEAR is available at https://github.com/juweipku/CLEAR.

4) Protocol: By strictly following the employed protocol in previous researches [22], [23], we evaluate the classification accuracy across the ten folds within the cross validation with LIBrary for Support Vector Machines (LIBSVM) [65]. Five runs of training and testing are conducted using different random seeds and the mean accuracy and standard deviation are reported in the results.

B. Performance Comparison

We summarize the quantitative results of different methods in Table II. According to the results, the following observations can be derived.

1) Most of the graph kernel methods and traditional unsupervised learning methods perform worse than graph CL methods. For instance, Graph2Vec performs worse than InfoGraph on six of eight datasets. Maybe the reason is that these approaches require hand-crafted designs and suffer from poor generalization. Instead, graph CL methods are able to extract more discriminative information from graph-structured data, showing the powerful representation learning ability of GNNs.
2) Among the previous state-of-the-art graph CL methods, CuCo achieves almost the best performance on most datasets. This is because CuCo can take better advantage of the effective negative sample selection and training strategies, which are prone to be effective and efficient. Specifically, Infograph, GraphCL, and JOAO are not as effective as CuCo on most datasets. Maybe the reason is that they typically require a large number of hard negative samples [66], and hard negatives are difficult to obtain by random sampling, which hinders these methods from learning discriminative representation, leading to poor performance.
3) Our framework CLEAR achieves the best performance on all seven datasets, which demonstrates the effectiveness of our framework. Specifically, the average improvement of our CLEAR over the best baseline CuCo is 3.17% on seven datasets. From the results, we deem that this improvement over state-of-the-art methods is mainly from both our effective semantic information exploration of local substructures via graph clustering as well as our multiview CL framework, which encourages consistency from both global and local views.

C. Ablation Study

To understand the contributions of different components in our model, we conduct ablation experiments to validate their effectiveness. In particular, we introduce the four model variants as follows.

1) *CLEAR-L:* We do not produce local-view graph representations and only contrast two global-view representations. Here, the subgraph augmentation strategy [24] is also involved.
2) *CLEAR-A:* We replace the attention mechanism with the average operation to aggregate both subgraph semantic information and subgraph interaction information.
3) *CLEAR-FM:* We do not model the subgraph interaction and only aggregate subgraph semantic information.
4) *CLEAR-LG:* The global-and-local contrastive loss (i.e., $\ell_m^{g\leftrightarrow l}$) is removed.

The results are recorded in Table III. We summarize the following findings: first, we can observe a consistent performance gain when comparing our full model with CLEAR-L on three datasets, illustrating the importance and necessity of our designs for modeling semantic information from a local view. The comparison also demonstrates that directly adopting a subgraph augmentation strategy [24] may lose some vital semantic information. Second, since different subgraph representations imply different semantic information and contribute differently to the final local-view representation, the results from the comparison of CLEAR-A and CLEAR indicate the effectiveness of the attention mechanism. Third, considering subgraph interaction can further improve model performance, as reflected by the better performance of CLEAR over CLEAR-FM. Fourth, CLEAR-LG shows worse performance compared with the full model, which, hence, illustrates

TABLE III

ABLATION STUDY OF SEVERAL MODEL VARIANTS (IN %). **BOLD** NUMBERS INDICATE THE BEST PERFORMANCE. THE RESULTS SHOW THE CONTRIBUTION OF DIFFERENT COMPONENTS IN THE PROPOSED FRAMEWORK

Methods	Datasets						
	PROTEINS	**DD**	**IMDB-B**	**IMDB-M**	**REDDIT-B**	**REDDIT-M-5k**	**COLLAB**
CLEAR-L	73.85 ± 0.62	76.96 ± 0.43	69.83 ± 0.69	47.46 ± 0.34	86.55 ± 1.15	54.01 ± 0.77	69.56 ± 0.38
CLEAR-A	75.13 ± 0.15	78.28 ± 0.21	71.89 ± 0.37	49.74 ± 0.36	88.69 ± 0.73	55.78 ± 0.46	71.08 ± 0.24
CLEAR-FM	74.44 ± 0.27	77.84 ± 0.25	71.45 ± 0.75	48.89 ± 0.21	87.90 ± 0.95	54.37 ± 0.56	71.06 ± 0.28
CLEAR-LG	74.29 ± 0.45	77.72 ± 0.20	70.07 ± 0.41	47.90 ± 0.59	88.38 ± 0.92	54.76 ± 0.67	69.71 ± 0.63
CLEAR (Ours)	**76.10 ± 0.35**	**79.37 ± 0.32**	**72.51 ± 0.81**	**50.26 ± 0.23**	**89.32 ± 1.27**	**56.73 ± 0.69**	**71.65 ± 0.33**

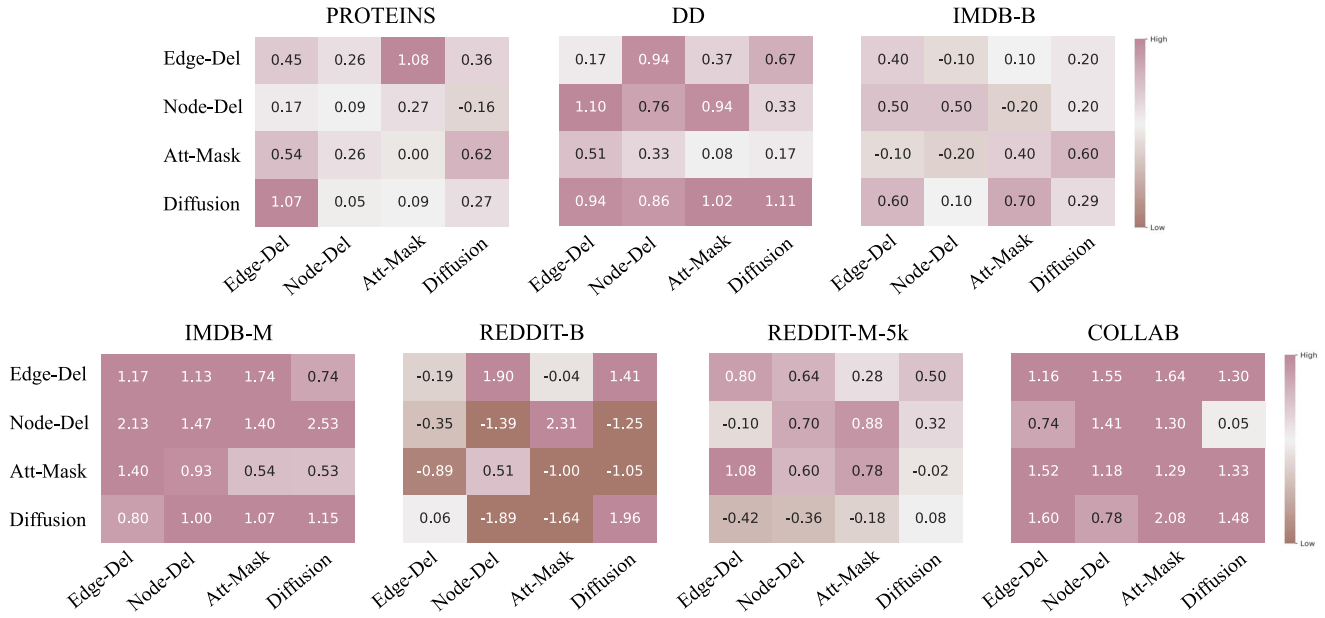

Fig. 5. Self-supervised representation learning accuracy gain (in %) when contrasting different augmented pairs, compared with training without any augmentation under seven datasets. Warmer colors indicate larger performance gains. The accuracies of baselines (without augmentation) are 75.02%, 78.18%, 71.80%, 47.73%, 88.15%, 55.77%, and 68.98% for the seven datasets, respectively.

that CL across local and global views can be beneficial to enhance the consistency and semantic-discriminative ability of graph-level representations. According to these findings, we can conclude that different model components indeed bring benefits to our framework and improve the performance.

D. Effect of Augmentation Strategies

As previously indicated, augmentation strategies are critical to graph CL. Therefore, in this part, we evaluate the impact of different data augmentation strategies on our CLEAR. The results of varying the augmentation strategies to produce $\hat{G}^1$ and $\hat{G}^2$ are shown in Fig. 5 with the following observations. First, it is clear that using graph augmentation strategies boosts the performance significantly when compared with the results without using augmentation in most cases. The explanation for this is that when proper augmentation strategies are adopted, the model is urged to learn representations invariant to corresponding noisy perturbations by maximizing the agreement between two augmented graphs. Second, some augmentation strategies may have a negative impact on performance. The potential reason is that a few augmentations are likely to change the semantics of graphs during the learning process, resulting in poor performance. Consequently, we randomly choose one of four augmentation strategies to tackle this potential issue in our implementation. Furthermore, we may use validation to choose target-invariant augmentation strategies on different datasets.

E. Parameter Sensitivity

In this section, we examine the sensitivity of the proposed CLEAR to various hyperparameters. Specifically, we investigate the effect of varying different embedding dimensions in hidden layers, different layer numbers, and different encoder architectures on representative benchmark datasets.

1) Effect of Different Embedding Dimensions: We begin by analyzing the influence of the embedding dimensions of hidden layers d on four representative datasets. It is known that the greater value of the embedding dimensions generally implies a larger model capacity. Therefore, we expect the model to perform well as dimensions increase. We fix other parameters to the ones which could achieve the best results and vary d in {8, 16, 32, 64, 128, 256} and show the results in Fig. 6. It can be seen that a larger embedding dimension mostly results in better accuracy before saturation. Nevertheless, too

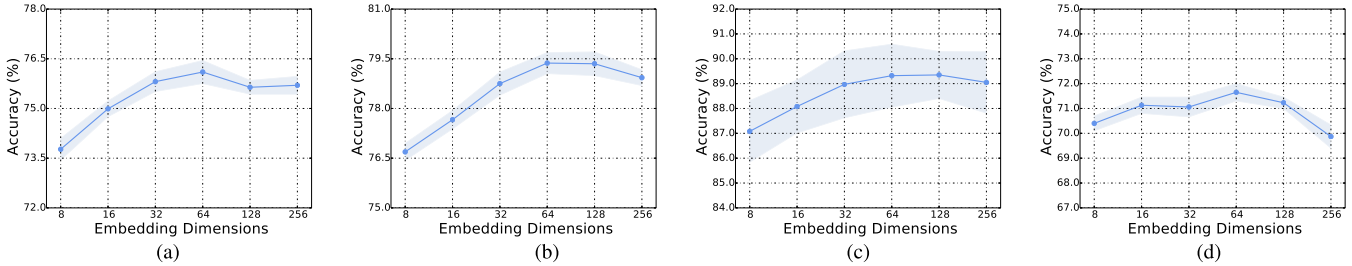

Fig. 6. Performance *w.r.t.* hidden dimension d. We can observe that model performance improves as the dimensions d grow before saturation in most cases. (a) PROTEINS. (b) DD. (c) REDDIT-B. (d) COLLAB.

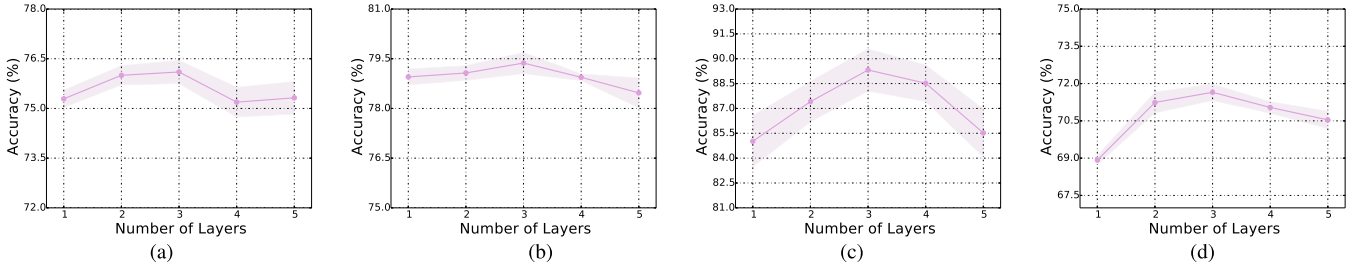

Fig. 7. Performance *w.r.t.* layer numbers K. We can observe that three propagation layers are sufficient to capture the structural semantics. (a) PROTEINS. (b) DD. (c) REDDIT-B. (d) COLLAB.

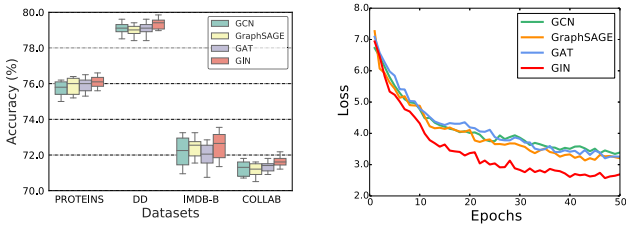

Fig. 8. Left: Performance with different encoder architectures. Right: Training curves of CLEAR on PROTEINS.

large dimensions could hurt the performance owing to overfitting resulting from the redundancy of parameters.

2) Effect of Different Layer Numbers: We next experiment with different model depths on four representative datasets to investigate whether our model can benefit from multiple embedding propagation. Specifically, the layer number is searched in the range of $\{1, 2, 3, 4, 5\}$. Fig. 7 shows the experiment results and we can observe that by increasing the depth of the GNN-based encoder from one to three, the performance on four datasets is improved. Generally, three propagation layers are sufficient to capture the structural semantics. The deeper layer may introduce noise and oversmoothing, leading to an inferior performance on downstream tasks.

3) Effect of Different Encoder Architectures: We study the effect of different encoder architectures in Fig. 8. We select four well-known GNNs, including GCN [5], GraphSAGE [67], GAT [34], and GIN [28]), and show the performance on four representative datasets. It can be observed that GIN outperforms the other three basic models on several datasets consistently, which verifies the efficacy of GIN with strong representation capacity. This also illustrates the reason why GIN is selected as the base model for GNN-based approaches. Finally, we plot the learning curves of four different encoder architectures on PROTEINS and find that GIN has the smallest training loss among four encoder architectures, which also demonstrates that GIN is able to fit the graph data very well. Also, we can see the empirical convergence of our framework.

TABLE IV
AVERAGE RUNNING TIME OF GRAPH CLUSTERING ALGORITHM (METIS) AND WHOLE EXECUTION IN A BATCH. THE LAST COLUMN SHOWS THE RATIO OF THE CLUSTERING TIME TO THE ENTIRE TIME

Datasets	Clustering Time	Whole Execution Time	Ratio
PROTEINS	0.32	2.20	0.15
DD	1.08	65.10	0.02
IMDB-B	0.27	1.46	0.19
IMDB-M	0.25	1.21	0.21
REDDIT-B	0.66	21.70	0.03
REDDIT-M-5k	0.76	21.93	0.03
COLLAB	0.71	6.27	0.11

F. Efficiency Analysis

As shown in Algorithm 1, the computing complexity of our method mainly depends on graph clustering and network propagation. We have analyzed that the complexity of network propagation in CLEAR is comparable with other CL methods. In this part, we need to figure out whether our extra graph clustering algorithms bring in a large cost to the whole framework. In Table IV, we present the average running time of graph clustering and the whole execution in a batch on all datasets. As a scalable and fast graph clustering library, Metis [44] is adopted for graph clustering. From Table IV, we observe that the graph clustering algorithm only takes a small portion of the whole execution time, demonstrating a small extra cost when applying such algorithms and their scalability on large datasets.

G. Case Study

In Fig. 9, we randomly select an example and visualize the attention distributions of two heads in the self-attention

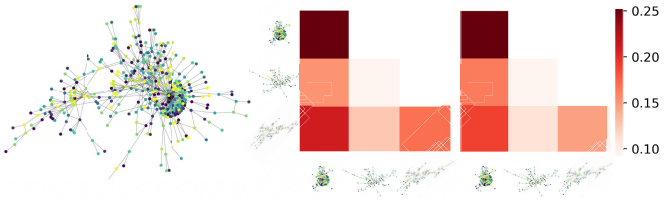

Fig. 9. Example of attention distributions in the self-attention interaction module. The left is the whole graph and the right shows heat maps to represent the attention values of two heads. The color scale represents the intensities of the weights, where a darker color indicates a higher value.

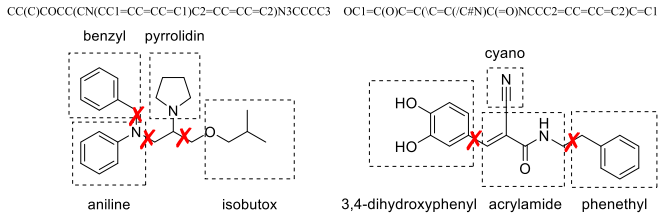

Fig. 10. Two molecular cases of graph clustering. Each dashed box represents a functional group with its name around and the red crosses partition each molecular into four or three clusters.

interaction module. Our graph clustering algorithm partitions the whole graph on the left into three subgraphs (i.e., the subgraph column in the middle), and the learned quantitative weights of two heads learned by our multihead self-attention mechanism are visualized on the right. The diagonal (nondiagonal) elements in matrices represent the attention weights for single (composite) subgraph representations. An interesting observation is that the first subgraph (i.e., the densest part in the whole graph) gets the biggest weight in both two heads, implying that the dense patches typically contain the most semantic information. Additionally, the composite subgraphs also get considerable weights in both two heads, indicating the necessity of modeling subgraph interactions to capture sufficient semantic information in local substructures. We further try other samples and find similar observations verifying the importance of dense substructures as well as substructure interactions in graph representation learning.

Moreover, we show two molecular cases in Tox21 datasets to study the benefits of graph clustering for representation learning. As shown in Fig. 10, we partition each molecule into four or three clusters. From the visualization, each cluster mostly corresponds to one or two functional groups. This validates that our graph clustering can explore the local information in these functional groups more efficiently with less perturbation of nodes from other groups, which benefits graph representation learning for downstream tasks.

H. Transfer Learning

This section is devoted to the empirical evaluation of the capability of our approach on large datasets. To achieve the goal, we apply CLEAR to transfer learning for predicting chemical molecule properties and biological protein functions. Following [69], we first pretrain the proposed model on a large-scale ZINC15 database and then fine-tune it on a range of open graph benchmark (OGB) [71] datasets to explore the transferability of the different pretraining schemes.

1) Datasets: To test the out-of-distribution performance of our method, we conduct experiments on seven OGB [71] molecule property prediction datasets to demonstrate its efficiency. For pretraining, we choose a subset of the ZINC15 database [72], [73] with two million unlabeled molecule graphs as in previous researches [69] using self-supervised learning methods. For fine-tuning, we adopt seven large-scale graph classification datasets in Moleculenet [74] to evaluate the transferability, with the same scaffold dataset split scheme [75] for a fair comparison.

2) Experiments Settings: Following [28], we adopt the GIN as the GNN-based encoder with 300 hidden units followed by a mean pooling readout function for pretraining. We adopt an Adam optimizer with the learning rate fixed to 10^{-3} to pretrain the GIN. An additional linear classifier is added on top of the pretrained encoder for fine-tuning and an Adam optimizer is adopted to train the model for 100 epochs. Tenfold cross validation is conducted to present the mean and the standard deviation of ROC-AUC scores over five times. Our CLEAR is compared with nonpretrain (with only fine-tuning after random initialization) and popular graph pretraining approaches. Specifically, besides graph CL approaches GraphCL [24] and JOAO [25], we select five representative pretraining techniques elaborated in detail as follows: 1) **EdgePred** [68]: It randomly removes part of edges while keeping all associated attributes and reconstructs the adjacent matrix. 2) **Infomax** [38]: It maximizes the mutual information between patch representations and their related graph-level representations created by a readout function. 3) **AttrMasking** [69]: It masks parts of node and edge attributes at random and then predicts their values on the basis of their neighbor information in the graph. 4) **ContextPred** [69]: It aims to utilize subgraphs to predict their structures and trains the GNN using negative sampling. 5) **GraphPartition** [70]: It presents a pretraining framework that partitions the nodes in a graph into approximately equal subsets while ensuring that the number of edges across different subsets is minimized.

3) Performance Analysis: In Table V, we compare the performance of the proposed CLEAR to various pretraining baselines under the transfer learning setting, From the results, we can find that our CLEAR achieves state-of-the-art performance by significantly outperforming the competing baselines on six of seven datasets. In particular, our approach can outperform the nonpretrain baseline by 12.5% on blood-brain barrier penetration (BBBP) and 16.0% on beta-secretase (BACE), which clarifies the efficacy of our multiview CL framework. The highest performance of each dataset is dispersed among competing baselines, demonstrating a considerable difference in the properties of different downstream task datasets. As a result, it is difficult to develop a generalized framework that captures the common knowledge of various datasets, while our method achieves the best results on most datasets, indicating local semantics are critical for learning graph semantics. Furthermore, in terms of average ROC-AUC, our approach performs better than the best pretraining strategy GraphPartition and CL approach JOAO, validating the superiority of our CLEAR.

4) Transfer Learning on Different Levels: Finally, we study the model performance of transfer learning at two different

TABLE V

RESULTS ON DOWNSTREAM MOLECULAR PROPERTY PREDICTION BENCHMARK DATASETS. THE REPORTED RESULTS ARE MEAN AS WELL AS THE STANDARD DEVIATION OF PREDICTION ACCURACY OVER FIVE TIMES USING 10-FOLD CROSS-VALIDATION. BOLD RESULTS INDICATE THE BEST PERFORMANCE. THE TRANSFERABILITY OF OUR PROPOSED CLEAR OUTPERFORMS ALL THE COMPETING BASELINES IN SIX OF SEVEN DATASETS

Methods	**Reference**	**Datasets**						
		BBBP	**Tox21**	**ToxCast**	**SIDER**	**MUV**	**HIV**	**BACE**
# Molecules		2039	7831	8575	1427	93087	41127	1513
# Binary prediction tasks		1	12	617	27	17	1	1
No Pre-Train	-	65.8 ± 4.5	74.0 ± 0.8	63.4 ± 0.6	57.3 ± 1.6	71.8 ± 2.5	75.3 ± 1.9	70.1 ± 5.4
EdgePred [68]	NeurIPS16 Workshop	67.3 ± 2.4	76.0 ± 0.6	64.1 ± 0.6	60.4 ± 0.7	74.1 ± 2.1	76.3 ± 1.0	79.9 ± 0.9
Infomax [38]	ICLR19	68.8 ± 0.8	75.3 ± 0.5	62.7 ± 0.4	58.4 ± 0.8	75.3 ± 2.5	76.0 ± 0.7	75.9 ± 1.6
AttrMasking [69]	ICLR20	64.3 ± 2.8	76.7 ± 0.4	**64.2 ± 0.5**	61.0 ± 0.7	74.7 ± 1.4	77.2 ± 1.1	79.3 ± 1.6
ContextPred [69]	ICLR20	68.0 ± 2.0	75.7 ± 0.7	63.9 ± 0.6	60.9 ± 0.6	75.8 ± 1.7	77.3 ± 1.0	79.6 ± 1.2
GraphPartition [70]	ICML20	70.3 ± 0.7	75.2 ± 0.4	63.2 ± 0.3	61.0 ± 0.8	75.4 ± 1.7	77.1 ± 0.7	79.6 ± 1.8
GraphCL [24]	NeurIPS20	69.7 ± 0.7	73.9 ± 0.7	62.4 ± 0.6	60.5 ± 0.9	69.8 ± 2.7	78.5 ± 1.2	75.4 ± 1.4
JOAO [25]	ICML21	70.2 ± 1.0	75.0 ± 0.3	63.0 ± 0.5	60.0 ± 0.8	71.7 ± 1.4	76.7 ± 1.2	77.3 ± 0.5
CLEAR (Ours)	Proposed	**71.0 ± 0.7**	**76.9 ± 0.5**	63.2 ± 0.3	**62.1 ± 0.6**	**76.7 ±1.6**	**78.9 ± 0.8**	**81.3 ± 0.7**

TABLE VI

PERFORMANCE OF TRANSFER LEARNING ACROSS DIFFERENT DATASETS

Methods	BBBP	MUV	HIV	BACE
JOAO	69.1 ± 0.5	73.8 ± 0.9	76.8 ± 1.2	79.2 ± 1.4
CLEAR (Ours)	69.8 ± 0.8	74.9 ± 1.1	77.0 ± 0.9	79.8 ± 1.5

TABLE VII

PERFORMANCE OF TRANSFER LEARNING ACROSS DIFFERENT TASKS

Methods	BBBP	MUV	HIV	BACE
JOAO	69.7 ± 1.1	74.4 ± 1.5	75.3 ± 0.8	77.1 ± 1.8
CLEAR (Ours)	70.2 ± 1.4	76.8 ± 1.3	76.7 ± 0.6	80.6 ± 1.3

levels, i.e., dataset level and task level. For the former, we pretrain and fine-tune the GNN with the same self-supervised representation learning task on the ZINC15 database and OGB datasets, respectively. The performance of representation learning is evaluated by training a linear classifier for downstream tasks. For the latter, we pretrain the network on OGB datasets with the self-supervised representation learning task and fine-tune it on the same dataset but with the classification task. The results are shown in Tables VI and VII. From the results, we have two observations. First, our CLEAR outperforms the representative baseline JOAO consistently on two tasks, which validates the effectiveness of our CLEAR. Second, compared to the results in Table VII, the performance in Table V which pretrains the network on large-scale ZINC15 datasets is better. The potential reason is that the model trained on the large-scale datasets is more generalized to benefit the performance in downstream tasks.

VI. CONCLUSION

In this article, we study self-supervised graph-level representation learning, which aims to learn the representations of an entire graph under the guidance of the data itself, and a novel approach called the CLEAR is presented. CLEAR is a unified framework, which models the structural semantics of a graph at graph-level and substructure-level granularities via graph clustering algorithms and graph augmentation strategies, respectively. Moreover, a multiview CL framework has been proposed to guide the model to learn consistent representations from different views, enhancing the semantic-discriminative ability of graph-level representations.

Comprehensive experiments on a variety of publicly available graph classification benchmark datasets and large-scale OGB datasets for transfer learning demonstrate the effectiveness of our proposed approach over a range of competitive baselines. In the future, we will explore the following directions: 1) further explore the large pretraining and transfer learning capabilities of the proposed method. 2) Extend the proposed method to more practical application scenarios, such as recommender systems, knowledge graphs, and molecular biology. 3) Theoretically analyze the use of contrastive training to improve the performance of graph-related tasks.

ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers for critically reading this article and for giving important suggestions to improve this article.

REFERENCES

[1] X. Chen, Y. Zhao, G. Liu, R. Sun, X. Zhou, and K. Zheng, "Efficient similarity-aware influence maximization in geo-social network," *IEEE Trans. Knowl. Data Eng.*, early access, Dec. 18, 2020, doi: 10.1109/TKDE.2020.3045783.

[2] Q. Wang *et al.*, "C-DeepTrust: A context-aware deep trust prediction model in online social networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 22, 2021, doi: 10.1109/TNNLS.2021.3107948.

[3] M. Zhang, L. Li, W. Hua, and X. Zhou, "Stream processing of shortest path query in dynamic road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 5, pp. 2458–2471, May 2022.

[4] M. Zeng, F. Zhang, F.-X. Wu, Y. Li, J. Wang, and M. Li, "Protein–protein interaction site prediction through combining local and global features with deep neural networks," *Bioinformatics*, vol. 36, no. 4, pp. 1114–1120, Sep. 2019.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2017. [Online]. Available: https://arxiv.org/abs/1609.02907

[6] W. Feng *et al.*, "Graph random neural network for semi-supervised learning on graphs," in *Proc. NeurIPS*, 2020, pp. 22092–22103.

[7] M. Stadler, B. Charpentier, S. Geisler, D. Zügner, and S. Günnemann, "Graph posterior network: Bayesian predictive uncertainty for node classification," in *Proc. NeurIPS*, 2021.

[8] Z. Wang, Y. Lei, and W. Li, "Neighborhood attention networks with adversarial learning for link prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3653–3663, Aug. 2021.

[9] L. Cai, J. Li, J. Wang, and S. Ji, "Line graph neural networks for link prediction," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, May 14, 2021, doi: 10.1109/TPAMI.2021.3080635.

[10] L. Cui, L. Bai, X. Bai, Y. Wang, and E. R. Hancock, "Learning aligned vertex convolutional networks for graph classification," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 10, 2022, doi: 10.1109/TNNLS.2021.3129649.

[11] H. Xie, J. Ma, L. Xiong, and C. Yang, "Federated graph classification over non-IID graphs," in *Proc. NeurIPS*, vol. 34, 2021, pp. 1–17.

[12] B. Jiang, K. Kloster, D. F. Gleich, and M. Gribskov, "AptRank: An adaptive PageRank model for protein function prediction on bi-relational graphs," *Bioinformatics*, vol. 33, no. 12, pp. 1829–1836, Jun. 2017.

[13] R. Kojima, S. Ishida, M. Ohta, H. Iwata, T. Honma, and Y. Okuno, "KGCN: A graph-based deep learning framework for chemical structures," *J. Cheminformatics*, vol. 12, no. 1, pp. 1–10, Dec. 2020.

[14] Z. Hao *et al.*, "ASGN: An active semi-supervised graph neural network for molecular property prediction," in *Proc. KDD*, 2020, pp. 731–752.

[15] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. AAAI*, 2018, pp. 4438–4445.

[16] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. NeurIPS*, 2018, pp. 1–16.

[17] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proc. ICML*, 2019, pp. 3734–3743.

[18] T. Zhang *et al.*, "Deep Wasserstein graph discriminant learning for graph classification," in *Proc. AAAI*, 2021, pp. 10914–10922.

[19] D. Duvenaud *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. NeurIPS*, 2015, pp. 1–9.

[20] X. Gao, W. Dai, C. Li, H. Xiong, and P. Frossard, "iPool–information-based pooling in hierarchical graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 31, 2021, doi: 10.1109/TNNLS.2021.3067441.

[21] A. D. Becke, "Perspective: Fifty years of density-functional theory in chemical physics," *J. Chem. Phys.*, vol. 140, no. 18, May 2014, Art. no. 18A301.

[22] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *Proc. ICLR*, 2020. [Online]. Available: https://arxiv.org/abs/1908.01000

[23] G. Chu, X. Wang, C. Shi, and X. Jiang, "CuCo: Graph representation with curriculum contrastive learning," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 1–7.

[24] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Proc. NeurIPS*, 2020, pp. 5812–5823.

[25] Y. You, T. Chen, Y. Shen, and Z. Wang, "Graph contrastive learning automated," in *Proc. ICML*, 2021, pp. 12121–12132.

[26] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, vol. 23, no. 1, pp. 31–42, Jan. 1976.

[27] Z. Liu, S. K. M. Nalluri, and J. F. Stoddart, "Surveying macrocyclic chemistry: From flexible crown ethers to rigid cyclophanes," *Chem. Soc. Rev.*, vol. 46, no. 9, pp. 2459–2478, 2017.

[28] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. ICLR*, 2019, pp. 1–16.

[29] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. 11th Annu. Conf. Comput. Learn. Theory (COLT)*, 1998, pp. 92–100.

[30] I. S. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.

[31] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 494–514, Feb. 2022.

[32] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proc. Web Conf.*, Apr. 2021, pp. 2069–2080.

[33] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2020.

[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, 2017. [Online]. Available: https://arxiv.org/abs/1710.10903

[35] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. ICML*, 2017, pp. 1263–1272.

[36] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. ICML*, 2020, pp. 1597–1607.

[37] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9729–9738.

[38] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *Proc. ICLR*, 2019. [Online]. Available: https://arxiv.org/abs/1809.10341

[39] D. Wu, F. Nie, X. Dong, R. Wang, and X. Li, "Parameter-free consensus embedding learning for multiview graph-based clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 29, 2021, doi: 10.1109/TNNLS.2021.3087162.

[40] S. E. Schaeffer, "Graph clustering," *Comp. Sci. Rev.*, vol. 1, no. 1, pp. 27–64, 2007.

[41] D. Shi, L. Zhu, Y. Li, J. Li, and X. Nie, "Robust structured graph clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4424–4436, Nov. 2020.

[42] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[43] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 11, pp. 1101–1113, Nov. 1993.

[44] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, Aug. 1999.

[45] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 257–266.

[46] K. Zhang, T. Li, S. Shen, B. Liu, J. Chen, and Q. Liu, "Adaptive graph convolutional network with attention graph clustering for co-saliency detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9050–9059.

[47] J. Klicpera, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," in *Proc. NeurIPS*, 2019, pp. 13366–13378.

[48] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *Proc. ICML*, 2020, pp. 4116–4126.

[49] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *Proc. ICLR Workshop Represent. Learn. Graphs Manifolds*, 2019, pp. 1–9.

[50] M. Meilă and W. Pentney, "Clustering by weighted cuts in directed graphs," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2007, pp. 135–144.

[51] A. Vaswani *et al.*, "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 1–11.

[52] S. Rendle, "Factorization machines," in *Proc. ICDM*, 2010, pp. 995–1000.

[53] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv:1807.03748*.

[54] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, pp. 47–56, Jun. 2005.

[55] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *J. Mol. Biol.*, vol. 330, no. 4, pp. 771–783, Jul. 2003.

[56] P. Yanardag and S. V. N. Vishwanathan, "Deep graph kernels," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 1365–1375.

[57] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Proc. AISTATS*, 2009, pp. 488–495.

[58] K. M. Borgwardt and H. Kriegel, "Shortest-path kernels on graphs," in *Proc. 5th IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2005, p. 8.

[59] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *J. Mach. Learn. Res.*, vol. 12, pp. 2539–2561, Sep. 2011.

[60] B. Adhikari, Y. Zhang, N. Ramakrishnan, and B. A. Prakash, "Sub2vec: Feature learning for subgraphs," in *Proc. PAKDD*, 2018, pp. 170–182.

[61] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," 2017, *arXiv:1707.05005*.

[62] B. Weisfeiler and A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9, pp. 12–16, 1968.

[63] L. Xia, C. Huang, Y. Xu, P. Dai, B. Zhang, and L. Bo, "Multiplex behavioral relation learning for recommendation via memory augmented transformer network," in *Proc. SIGIR*, 2020, pp. 2397–2406.
[64] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015. [Online]. Available: https://arxiv.org/abs/1412.6980
[65] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.
[66] Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus, "Hard negative mixing for contrastive learning," in *Proc. NeurIPS*, 2020, pp. 21798–21809.
[67] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NeurIPS*, 2017, pp. 1025–1035.
[68] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.
[69] W. Hu *et al.*, "Strategies for pre-training graph neural networks," 2019, *arXiv:1905.12265*.
[70] Y. You, T. Chen, Z. Wang, and Y. Shen, "When does self-supervision help graph convolutional networks?" in *Proc. ICML*, 2020, pp. 10871–10880.
[71] W. Hu *et al.*, "Open graph benchmark: Datasets for machine learning on graphs," 2020, *arXiv:2005.00687*.
[72] A. Mayr, "Large-scale comparison of machine learning methods for drug target prediction on ChEMBL," *Chem. Sci.* vol. 9, no. 24, pp. 5441–5451, 2018.
[73] A. Gaulton *et al.*, "ChEMBL: A large-scale bioactivity database for drug discovery," *Nucleic Acids Res.*, vol. 40, no. D1, pp. D1100–D1107, 2012.
[74] Z. Wu *et al.*, "MoleculeNet: A benchmark for molecular machine learning," *Chem. Sci.*, vol. 9, no. 2, pp. 513–530, 2018.
[75] B. Chen, R. P. Sheridan, V. Hornak, and J. H. Voigt, "Comparison of random forest and pipeline pilot Naïve Bayes in prospective QSAR predictions," *J. Chem. Inf. Model.*, vol. 52, no. 3, pp. 792–803, Mar. 2012.

Xiao Luo received the B.S. degree in mathematics from Nanjing University, Nanjing, China, in 2017. He is currently pursuing the Ph.D. degree with the School of Mathematical Sciences, Peking University, Beijing, China.

His research interests include machine learning on graphs, image retrieval, statistical models, and bioinformatics.

Wei Ju (Graduate Student Member, IEEE) received the B.S. degree in mathematics from Sichuan University, Sichuan, China, in 2017. He is currently pursuing the Ph.D. degree in computer science from Peking University, Beijing, China.

His research interests include machine learning on graphs, including graph neural networks and graph representation learning.

Meng Qu received the B.Sc. degree in computer science from Peking University, Beijing, China, in 2015, and the M.Sc. degree in computer science from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2018. He is currently pursuing the Ph.D. degree with the Mila—Quebec AI Institute, Montreal, QC, Canada, with a focus on reasoning on graph-structured data, such as knowledge graphs, under the supervision of Prof. Jian Tang.

He has authored or coauthored several papers on combining deep learning and statistical relational learning for knowledge reasoning in top-tier venues.

Yiyang Gu received the B.S. degree in computer science from Peking University, Beijing, China, in 2021, where he is currently pursuing the Ph.D. degree in computer science.

His research interests include graph representation learning, knowledge graph, and bioinformatics.

Chong Chen (Member, IEEE) received the B.S. degree in mathematics and applied mathematics and the Ph.D. degree in statistics from Peking University, Beijing, China, in 2013 and 2019, respectively.

He is currently a Staff Algorithm Engineer with the Discovery, Adventure, Momentum and Outlook (DAMO) Academy, Alibaba Group, Hangzhou, China. His research interests include statistics, machine learning, and computer vision.

Minghua Deng received the B.S. and Ph.D. degrees from Peking University, Beijing, China, in 1991 and 1998, respectively.

He is currently a Professor of statistics with the School of Mathematical Sciences, Peking University. His research interests include statistical models, genetics, and bioinformatics.

Xian-Sheng Hua (Fellow, IEEE) received the B.S. and Ph.D. degrees in applied mathematics from Peking University, Beijing, China, in 1996 and 2001, respectively.

In 2001, he joined Microsoft Research Asia, Beijing, as a Researcher. He has been a Senior Researcher with Microsoft Research Redmond, Redmond, WA, USA, since 2013. He became a Researcher and the Senior Director with the Alibaba Group, Hangzhou, China, in 2015. He has authored or coauthored over 250 research papers and has filed over 90 patents. His research interests include the areas of multimedia search, advertising, understanding, and mining, pattern recognition, and machine learning.

Dr. Hua is an ACM Distinguished Scientist. He has served on the Technical Directions Board of the IEEE Signal Processing Society. He was a recipient of the MIT Technology Review Innovators Under 35 Asia Pacific (MIT35). He served as the Program Co-Chair for the ACM Multimedia 2012, the IEEE International Conference on Multimedia and Expo (ICME) 2012, and the IEEE ICME 2013.

Ming Zhang received the B.S., M.S., and Ph.D. degrees in computer science from Peking University, Beijing, China, in 1988, 1991, and 2005, respectively.

She is currently a Full Professor with the School of Computer Science, Peking University. She has authored or coauthored more than 200 research papers on text mining and machine learning in the top journals and conferences. She is a leading author of several textbooks on data structures and algorithms in Chinese and the corresponding course is awarded as the National Elaborate Course, National Boutique Resource Sharing Course, National Fine-designed Online Course, and National First-Class Undergraduate Course by Ministry of Education (MOE) China.

Prof. Zhang is a member of the Advisory Committee of the Ministry of Education in China. She is one of the 15 members of the ACM/IEEE CC2020 Steering Committee. She received the Best Paper Award of the International Conference on Machine Learning (ICML) 2014 and the Best Paper Nominee of World Wide Web (WWW) 2016. She is the Chair of the ACM Special Interest Group of Computer Science Education (SIGCSE) China.