Self-supervised Graph-level Representation Learning with Adversarial Contrastive Learning

XIAO LUO, Department of Computer Science, University of California, Los Angeles, USA
WEI JU, YIYANG GU, ZHENGYANG MAO, and LUCHEN LIU, School of Computer Science, Peking University, China
YUHUI YUAN, Microsoft Research Asia, Beijing, China
MING ZHANG, School of Computer Science, Peking University, China

The recently developed unsupervised graph representation learning approaches apply contrastive learning into graph-structured data and achieve promising performance. However, these methods mainly focus on graph augmentation for positive samples, while the negative mining strategies for graph contrastive learning are less explored, leading to sub-optimal performance. To tackle this issue, we propose a Graph Adversarial Contrastive Learning (GraphACL) scheme that learns a bank of negative samples for effective self-supervised whole-graph representation learning. Our GraphACL consists of (i) a graph encoding branch that generates the representations of positive samples and (ii) an adversarial generation branch that produces a bank of negative samples. To generate more powerful hard negative samples, our method minimizes the contrastive loss during encoding updating while maximizing the contrastive loss adversarially over the negative samples for providing the challenging contrastive task. Moreover, the quality of representations produced by the adversarial generation branch is enhanced through the regularization of carefully designed bank divergence loss and bank orthogonality loss. We optimize the parameters of the graph encoding branch and adversarial generation branch alternately. Extensive experiments on 14 real-world benchmarks on both graph classification and transfer learning tasks demonstrate the effectiveness of the proposed approach over existing graph self-supervised representation learning methods.

CCS Concepts: • **Information systems** → **Data mining;**

Additional Key Words and Phrases: Graph representation learning, graph neural networks, contrastive learning, self-supervised learning

ACM Reference format:
Xiao Luo, Wei Ju, Yiyang Gu, Zhengyang Mao, Luchen Liu, Yuhui Yuan, and Ming Zhang. 2023. Self-supervised Graph-level Representation Learning with Adversarial Contrastive Learning. *ACM Trans. Knowl. Discov. Data.* 18, 2, Article 34 (November 2023), 23 pages.
https://doi.org/10.1145/3624018

X. Luo and W. JU contributed equally to this research with order determined by flipping a coin.
This article is partially supported by the National Natural Science Foundation of China with Grants No. 62276002 and 62106008 as well as the China Postdoctoral Science Foundation with Grant No. 2023M730057.
Authors' addresses: X. Luo, Department of Computer Science, University of California, Los Angeles, USA, 90095; e-mail: xiaoluo@cs.ucla.edu; W. Ju, Y. Gu, Z. Mao, L. Liu, and M. Zhang (Corresponding author), School of Computer Science, Peking University, Beijing, China, 100871; e-mails: {juwei, yiyanggu}@pku.edu.cn, mao.zhengyang.cn@gmail.com, liuluchen@pku.edu.cn, mzhang_cs@pku.edu.cn; Y. Yuan (Corresponding author), Microsoft Research Asia, Beijing, China, 100080; e-mail: yuyua@microsoft.com.

1556-4681/2023/11-ART34
https://doi.org/10.1145/3624018

1 INTRODUCTION

Recently, graphs have become an effective tool to depict diverse types of non-Euclidean data including traffic networks [42, 43, 80] and social networks [33, 70]. Due to their versatility, graph-like data structures may capture a great deal of information, which is important in a wide range of domains. Extensive efforts have been made covering a range of graph machine learning tasks including node classification [17, 40], graph classification [35, 47], link prediction [63], and graph alignment [29, 91]. Inspired by the progress of deep learning, the center of graph machine learning has become graph representation learning, which can mainly be classified into learning node representations and graph representations. The former produces dense vector embeddings for individual nodes in a network while the latter predicts discriminative representations of whole graphs, which is under-explored. Such a task can benefit various downstream applications, including social network analysis [81, 88] and molecule property prediction [27].

In the past few years, **graph neural networks (GNNs)** have achieved superior performance in graph-level representation learning [26, 87]. They are capable of embedding graphs with different sizes into low-dimensional representations via iterative message passing and summarization procedures [19, 25, 87]. In particular, for each node, we aggregate the neighborhood information from its neighbors to update the node representation in an iterative manner. Eventually, a summarization function is utilized to produce a graph-level representation integrating all node representations. Thus, the acquired graph-level representation would convey graph structural semantics to benefit a range of downstream applications.

In spite of their success, traditional GNN methods fail to optimize well without a large number of labeled graphs. Unfortunately, it is often prohibitively costly to acquire labeled data in many fields, such as biochemistry [27]. The labeled graphs could be scarce while the unlabeled graphs are easy to collect in practice. It is therefore highly promising to learn graph-level representations in an unsupervised manner. Recently, self-supervised contrastive learning has made significant progress in computer vision [7, 28] and recommender systems [36, 54, 71]. Inspired by this, many recent works [8, 60, 82, 83] have brought this technique to unsupervised graph representation learning. These approaches enforce a graph to get comparable representations to its augmented view in comparison to other samples, yielding discriminative graph representations to facilitate a variety of downstream applications.

However, two inherent issues must be carefully explored when applying contrastive learning to graphs: (i) data augmentations to produce congruent, semantic-preserving samples for each graph and (ii) negative mining strategies to provide challenging contrastive learning tasks. The success of contrastive learning on graphs is largely determined by these two elements. The majority of recent works focus on the first condition and propose various graph augmentation operations, such as randomly discarding a certain portion of nodes or edges in the whole graph and further developing automatic augmentation strategies [64, 82, 83]. However, negative mining strategies have drawn little attention and exploration, where informative and high-quality negative samples are a critical guarantee for the success of optimization.

In recent graph contrastive learning, there are two types of algorithms to construct negative instances. Inspired by **Momentum Contrast (MoCo)** [28], the first type of algorithm (e.g., **Curriculum Contrastive learning (CuCo)** [8]) maintains an explicit queue of negative samples from previous mini-batches following a First-In-First-Out scheme. Nevertheless, since only a tiny part of the graph representations in the queue will be updated, it would be hard to keep up with the fast change in graph representations over iterations. Additionally, these methods usually adopt the momentum update scheme to stabilize the queue updating, further hindering the track of the negative representations in the bank. Hence, the GNN-based encoder would be trained inefficiently, since a partly updated queue of negative instances may be biased and fail to cover enough pivotal

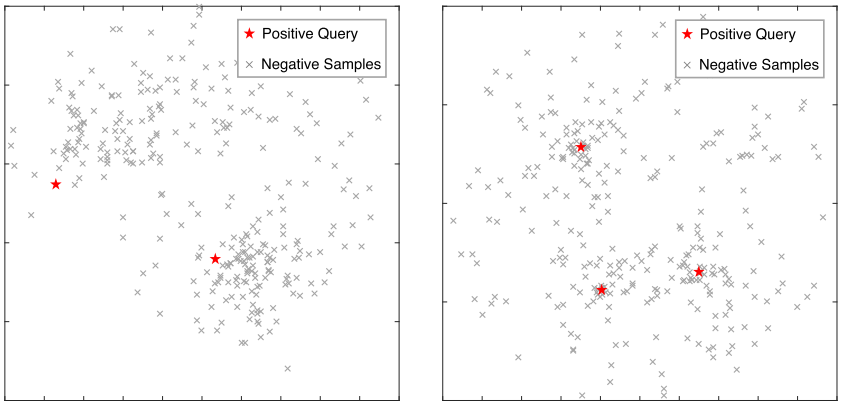


Fig. 1. Illustration of conventional methods and our GraphACL. Left: In conventional methods, negative samples may be biased and uninformative (i.e., far from a positive query). Right: Our GraphACL can generate both informative and high-quality negative samples for effective contrastive learning.

challenging samples that need to be separated from positive queries as illustrated in Figure 1. The second type (e.g., GraphCL and JOAO [82, 83]) employs the contrastive learning framework of SimCLR [7], which frees up the need for a separate queue of negative samples. Instead, the framework contrasts the positive queries against the other samples in the minibatch. Despite their simplicity, it throws away negative instances from earlier batches, therefore necessitating a high batch size to provide enough instructive negative examples for contrastive learning. This type of algorithm requires excessive memory and computational costs, which are typically unaffordable for existing graph contrastive learning techniques [82, 83]. As such, it is necessary to develop effective negative mining strategies for graph contrastive learning, which is usually under-explored in existing methods. However, this issue is non-trivial due to the following major challenges: (i) We need to produce informative hard negative samples to discriminate against positive queries, and (ii) the quality of generated negative samples needs to be promised (e.g., globally large diversity to avoid collapsing into trivial solutions).[1]

To address the aforementioned crucial challenges, we present a novel framework, **Graph Adversarial Contrastive Learning (GraphACL)**, for unsupervised graph-level representation learning, which leverages adversarial learning to actively optimize negative samples. In particular, apart from introducing a graph encoding branch that includes various augmentations for constructing positive sample pairs, our GraphACL consists of an adversarial generation branch to optimize representations of a bank of negative samples. Specifically, we overcome the above challenges by (i) maximizing the contrastive loss in an adversarial fashion and (ii) designing a combination of bank divergence loss and bank orthogonality loss to avoid trivial solutions and reduce the redundancy of negative samples. The two branches are trained alternately by iterative optimization. On the one side, the graph encoding branch is updated by reducing the contrastive loss as in classic graph contrastive learning methods. On the other side, the adversarial generation branch is updated by minimizing two bank losses while simultaneously maximizing the contrastive learning loss in an adversarial fashion, which encourages the negatives to approach the query graphs in the batch. We also extend our model into an asymmetric architecture, which can prevent potential collapsed solutions where all the network outputs are identical. Extensive experiments are con-

[1]Note that informativeness and large diversity are not contradictory, since we will optimize a batch of different positive samples during training. In fact, the negatives should be diverse globally, but some of them center locally around each positive query.

ducted on 14 well-known benchmark datasets to verify the efficacy of our proposed methods for both the graph classification task and the transfer learning task. The results show that GraphACL achieves state-of-the-art performance over competing baselines. The contributions of this work can be summarized as follows:

— We make the first attempt toward actively training negative samples for effective graph contrastive learning and propose a novel framework GraphACL for unsupervised graph-level representation learning.
— We employ adversarial learning and two bank losses to produce informative and high-quality negative samples to discriminate against positive samples.
— We analyze the benefit of three loss objectives from various views to show how our adversarial updating of negative representations helps our graph contrastive learning.
— Extensive experiments on a variety of popular graph classification and transfer learning datasets demonstrate the superiority of the proposed approach GraphACL.

2 RELATED WORK

2.1 Graph Representation Learning

GNNs have emerged as an effective approach that aims to extend the deep neural networks to handle arbitrary graph-structured data [33, 73, 89] and have been widely used for a large number of applications such as edge classification [17], graph alignment [91], traffic forecasting [42, 43], node prediction [20], and so on. Existing GNN methods typically adopt an iterative message passing process [16, 19] to recursively learn structural semantics. Nowadays, it is well recognized that an effective and informative representation of the whole graph is vital to the learning performance of graph machine learning models in a variety of domains and tasks. The representation of the whole graph can be obtained through neighbor propagation via graph neural networks and global summarizing, for example, by averaging the representation vectors of all nodes in the graph [13, 56, 61, 67]. Despite their effectiveness, these GNN algorithms cannot be optimized without a huge number of expensive and limited labeled graphs [27]. To release the training cost, our work aims to handle unsupervised/self-supervised graph-level representation learning task and studies the challenge of negative sample construction and further proposes a novel graph contrastive learning framework GraphACL.

2.2 Graph Contrastive Learning

Contrastive learning (CL) explores underlying semantics between contrastive pairs obtained by random augmentation of the original samples [7]. Hadsell et al. [23] is the pioneering work to learn representations by contrasting positive pairs against negative pairs. A variety of pretext works are based on diverse forms of the contrastive loss function, which has a connection with the exemplar-based task and noise-contrastive estimation [11]. Nowadays, MoCo [28] proposes to construct a dynamic dictionary and a queue, which enable sufficient information from negative samples for effective contrastive learning. SimCLR [7] further simplifies the learning paradigm without using a memory bank. CL has been enhanced with a range of techniques, which can benefit various tasks such as clustering [14], action recognition [77], and recommender systems [44, 90]. For example, Xu et al. [76] introduce an adaptive augmentation strategy for effective skeleton-based action recognition. Shu et al. [59] introduce anchor graphs for effective contrastive learning in fine-grained scenarios. Further works incorporate squeezing techniques [78] and hierarchical views [75] to improve the representation learning.

Extensive attempts have been made to incorporate CL with graph neural networks [34, 45, 46, 55, 64]. As an early work, InfoGraph [60] attempts to increase the mutual information between

Table 1. Summary of Notations and Descriptions

Notations	Descriptions
$\mathcal{G}_i$	Graph sample
$\hat{\mathcal{G}}_i, \hat{\mathcal{G}}_i'$	Augmented graph samples
t, t'	Random augmentation functions
$\mathbf{z}_i, \mathbf{z}_i'$	Graph-level representations
g_θ	Graph encoder
g_ϕ	Momentum graph encoder
$\mathcal{N}$	Negative sample bank
J	The number of negative samples
$\boldsymbol{n}_j$	Generated negative sample
τ	Temperature parameter
$\mathbf{N}$	Negative representation matrix
f_l	Negative dimension vector

global representations and substructure representations, respectively. **Rationale-aware Graph Contrastive Learning (RGCL)** [45] introduces rationales into graph contrastive learning for effective discrimination learning. Recent efforts have been made toward graph-level representation learning. These methods [8, 82, 83, 86] mostly follow the scheme of SimCLR and MoCo, which expect a graph to share similar embeddings to its augmented views in comparison to those from other graphs. These methods propose various graph augmentation strategies but negative mining strategies are typically underexplored. Hence, these approaches fail to construct discriminative negative samples for efficient graph contrastive learning while our work better explores negative mining strategies through adversarial learning.

2.3 Adversarial Learning

Adversarial learning has been widely used to enhance model generalization with a range of applications in computer vision [21]. For example, Mardy-AT [48] combines adversarial learning with robust optimization, which improves the performance with it comes to adversarial attacks. Further, a sufficient evaluation of different tricks in adversarial training is conducted in Reference [53]. Some works also integrate adversarial learning into a self-supervised learning framework, producing robust image representations [30] for visual tasks. Recently, extensive efforts are adopted to extend this technique into graph machine learning [9, 15]. For example, GraphGAN [66] generates fake node pairs or triplets to confuse the graph neural networks to enhance the model generalization. However, these works typically focus on enhancing the model robustness for node-level tasks. By contrast, we adopt adversarial learning to generate a bank of negative samples, providing the challenging contrastive task for learning discriminative graph-level representations. Moreover, since adversarial learning may suffer from instability such as mode collapse [37], we further introduce two carefully designed regularization losses to improve the quality of generated negatives.

3 PRELIMINARIES AND NOTATIONS

In this part, we begin with an introduction to formal definitions and notations listed in Table 1 for clarity and then formalize our problem definition.

Definition 1 (Graph). A graph is denoted as $\mathcal{G} = (V, E, \mathbf{X})$, where V and E represent the set of vertex and edges, respectively. Let $\mathbf{x}_v$ represent the attribute vector of $v \in V$, and $\mathbf{X} \in R^{|V| \times d_0}$ denotes the node attribute matrix, in which d_0 denotes the dimension of the attribute vector.

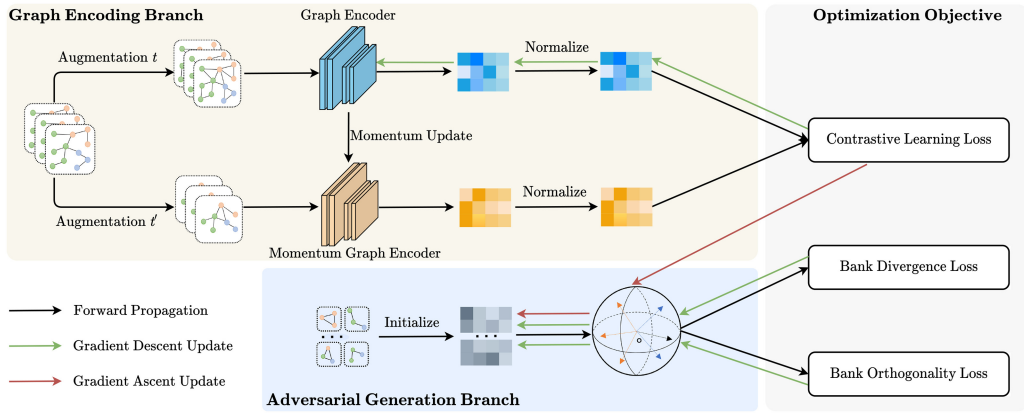


Fig. 2. Illustration of our GraphACL scheme. We use a graph encoding branch parameterized by two graph encoders to generate the representations of positive samples and an adversarial generation branch produces a bank of negative samples. The first branch is optimized by minimizing graph contrastive learning loss, and the second branch is optimized by maximizing contrastive learning loss while simultaneously minimizing two bank regularization losses.

Definition 2 (Unsupervised Graph Representation Learning). We have access to M unlabeled graphs $\{\mathcal{G}_1, \ldots, \mathcal{G}_M\}$. The purpose is to learn a graph encoder, which maps each graph $\mathcal{G}_m$ into an embedding vector $\mathbf{z}_m$, among which we expect these graph embeddings are suitable for various downstream tasks including graph classification.

4 OUR APPROACH

4.1 Framework Overview

This article introduces the GraphACL to actively learn a bank of negative samples for unsupervised graph representation learning. Negative mining strategies are underexplored in graph contrastive learning techniques. Prior works typically maintain a queue of negative representations or treat other graphs as negatives from the current minibatch. However, these methods usually suffer from biased and uninformative negative samples, leading to the ineffectiveness of contrastive learning.

To tackle the issue, our approach leverages the idea of adversarial learning to actively generate negative samples. In particular, our proposed GraphACL consists of a graph encoding branch to produce graph-level representations for positive samples and an adversarial generation branch to optimize a bank of negative samples for contrastive learning. For effective negative mining, we not only maximize the contrastive loss in an adversarial fashion for more hard negatives but also minimize two carefully designed bank regularization losses to reduce the redundancy of negatives. Finally, we optimize the framework in an alternative manner. On the one hand, following conventional graph contrastive learning techniques, the graph encoding branch is updated by minimizing the contrastive learning to pull close queries and their positives compared with negatives in the embedding space. On the other hand, the adversarial generation branch is trained by minimizing two regularization losses while concurrently maximizing the contrastive learning loss in an adversarial way, allowing negative examples to keep track of query samples in the minibatch. More details can be found in Figure 2.

The rest of this section is structured as below. We introduce the graph encoding branch with graph encoder and graph augmentation strategies in Section 4.2. Then, we review the contrastive learning framework and introduce an adversarial generation branch in detail in Section 4.3. Further,

we formulate the adversarial learning framework in Section 4.4. Finally, we analyze the strengths of our proposed GraphACL in Section 4.5.

4.2 Graph Encoding Branch

In the graph encoding branch, each graph first goes through two types of augmentation followed by two different graph encoders, producing novel rational graph-level representation pairs for contrastive learning.

GNN-based Encoder. Graph representation learning aims to project graph samples into an embedding space. GNNs have recently shown their superior capability for modeling various graph-structured data. Hence, we primarily adopt GNNs as graph encoders in our branch. Generally, GNNs adopt the message passing process to encode the topological semantics into node embeddings [19]. In particular, for each node v of given graph $\mathcal{G} = (V, E, \mathbf{X})$, first we aggregate the embeddings of all its neighbors at the $(k-1)$-th layer. Let $\mathcal{S}(v)$ denote the neighborhood of node v, and the embedding of v at the kth layer $\mathbf{h}_v^{(k)}$ is updated as follows:

$$\mathbf{h}_v^{(k)} = \mathrm{COM}_\theta^{(k)}\left(\mathbf{h}_v^{(k-1)}, \mathrm{AGG}_\theta^{(k)}\left(\left\{\mathbf{h}_u^{(k-1)}\right\}_{u\in\mathcal{S}(v)}\right)\right), \tag{1}$$

where $\mathrm{AGG}_\theta^{(k)}$ and $\mathrm{COM}_\theta^{(k)}$ denote the aggregation and combination functions parameterized by θ at the kth layer, respectively. Eventually, for graph-level representation learning tasks, we utilize $\mathrm{READOUT}(\cdot)$ to summarize updated node embeddings at the final layer. The final graph-level representation is written as

$$g_\theta(\mathcal{G}) = \mathrm{READOUT}\left(\left\{\mathbf{h}_v^{(K)}\right\}_{v\in V}\right), \tag{2}$$

where READOUT is a permutation-invariant approach such as averaging. Here we use the sum operation for READOUT following Reference [79], because it is capable of providing crucial information regarding the graph size.

Graph Augmentations. Data augmentation is essential for contrastive learning, which generates rational data with semantics unchanged mostly [7]. In our method GraphACL, we adopt four common strategies to perturb the topology and attributes of a graph as follows [83]. (1) *Edge dropping*: This randomly drops part of the edges in a graph. The prior of this strategy is that removing some nodes typically cannot change the property of the graph. (2) *Node dropping*: This randomly chooses several nodes and drops them from the graph, together with their connected edges. This strategy presumes that the semantic of the graph is typically robust to the edge connectivity mode variances. (3) *Attribute masking*: This randomly samples some vertices and then masks their partial feature attributes. The underlying prior is that missing some of the node attributes does not seriously influence the semantics of the graph. (4) *Subgraph sampling*: This constructs a subgraph from a graph via a random walk strategy. The rationale behind this strategy is that the semantics of the graph can be retained in its local compact structure. One potential advantage is that subgraph may require smaller GPU memory in practice.

For each G, we begin with generating two correlated views $\hat{\mathcal{G}}_i = t(\mathcal{G}_i), \hat{\mathcal{G}}_i' = t'(\mathcal{G}_i)$ by randomly selecting one of the four augmentation strategies for each view, where t and t' denote two stochastic augmentation functions. Inspired by MoCo [28], we use two GNN-based encoders (i.e., graph encoder g_θ and momentum graph encoder g_ϕ with the same architecture) to extract novel rational graph vector pair $\{\mathbf{z}_i, \mathbf{z}_i'\}$ for augmented graphs $\hat{\mathcal{G}}_i$ and $\hat{\mathcal{G}}_i'$ as follows:

$$\mathbf{h}_i = g_\theta(\hat{\mathcal{G}}_i), \mathbf{h}_i' = g_\phi(\hat{\mathcal{G}}_i'). \tag{3}$$

Finally, two projection heads $\psi_\theta(\cdot)$ and $\psi_\theta(\cdot)$ are utilized to map these graph vectors into another embedding space for final graph representations, i.e., $\mathbf{z}_i = \psi_\theta(\mathbf{h}_i)$ and $\mathbf{z}_i' = \psi_\theta(\mathbf{h}_i')$.

4.3 Adversarial Generation Branch

We first review the objective in graph contrastive learning methods [82, 83]. To maximize the similarity between each positive pair $\{\mathbf{z}_i, \mathbf{z}'_i\}$ in comparison to a bank of negative samples $\mathcal{N} = \left\{\mathbf{n}_j \mid j = 1, \ldots, J\right\}$, the noise-contrastive estimation loss [52] adopted is a softmax form,

$$\mathcal{L}_{CL} = -\frac{1}{M}\sum_{i=1}^{M} \log \frac{e^{\mathbf{z}_i \star \mathbf{z}'_i/\tau}}{e^{\mathbf{z}_i \star \mathbf{z}'_i/\tau} + \sum_{j=1}^{J} e^{\mathbf{z}_i \star \mathbf{n}_j/\tau}}, \tag{4}$$

where τ is a positive temperature parameter set to 0.5 following References [82, 83] and $\mathbf{z}_i \star \mathbf{z}'_i$ denotes the cosine similarity of $\mathbf{z}_i$ and $\mathbf{z}'_i$. Here J denotes the number of negatives. Following Reference [8], we use the dot product to replace the cosine similarity metric function in Equation (4) for calculation simplification.

These negative examples play an important role in graph contrastive learning and require sophisticated design. Existing graph contrastive learning methods typically either construct a queue of negative representations iteratively updated in a queue strategy or treat other graphs in the batch as negative samples. These negative mining strategies cannot produce challenging negative samples for graph contrastive learning, hindering the encoder from producing discriminative graph-level representations. To address this issue, we propose to actively train a bank of negative samples for the whole dataset instead [30]. Specifically, GraphACL introduces a negative generation branch that is trained against the graph encoding branch in an adversarial manner. The adversarial objective is formulated in a minimax form as follows:

$$\theta^\star, \mathcal{N}^\star = \arg\min_{\theta} \max_{\mathcal{N}} \mathcal{L}_{CL}(\theta, \mathcal{N}), \tag{5}$$

where the representations of positive samples are produced by two graph encoders g_θ and g_ϕ. From Equation (5), we can observe that the graph encoding branch and the adversarial generation branch are two mutually interacted. On the one hand, the adversarial generation branch is trained to confuse each positive pair, providing informative negative samples for contrastive learning. On the other hand, the graph encoding branch is trained to generally enhance the discrimination capacity by distinguishing the positive queries from the bank of adversarial negative samples.

Inspired by References [30, 51], we seek to introduce "hidden graphs" to parameterize negative samples. However, parameterized large-size graphs are unaffordable for framework optimization. Instead, we parameterize their dense representations of negative samples[2] with free variables, which are all learnable vectors with l_2 norm equal to 1 for the convention of contrastive learning. However, due to the large freedom of negative representations, their quality is hard to promise. For instance, when the negative samples lack diversity, the positive sample pair may not be pulled close and the representation collapse may even occur, leading to trivial outputs [69]. The detailed explanation will be shown in Section 4.5. To tackle this issue, we design two bank losses to regularize trainable negative representations.

Bank Divergence Loss. Intuitively, the low diversity in the negative representation bank not only brings in much redundancy but also impacts the optimization process of contrastive learning.[3] Hence, we propose a bank divergence loss objective to penalize the solution when two negative representations are almost identical,

$$\mathcal{L}_{BD} = -\sum_{j=1}^{J} \log\left(\frac{\exp\left(\boldsymbol{n}_j^\top \boldsymbol{n}_j/\tau\right)}{\sum_{j'=1}^{J}\exp\left(\boldsymbol{n}_j^\top \boldsymbol{n}_{j'}/\tau\right)}\right). \tag{6}$$

[2]It would be difficult to reconstruct the graph structures considering their high complexity.
[3]The detailed explanation will be shown in Section 4.5.

From the above Equation (6), we can observe that minimizing $\mathcal{L}_{BD}$ equals to minimizing $\boldsymbol{n}_j^\top \boldsymbol{n}_{j'} (j \neq j')$, which enlarges the divergence of the negative representation bank to avoid the redundancy of negative representations. The bank divergence loss has a similar form to contrastive loss, but all graph representations in loss are from negative samples. In particular, the loss increases the distance between different representations by encouraging them to approximate a uniform distribution in the embedding space [68].

Bank Orthogonality Loss. Furthermore, a high correlation of negative representations is also likely to harm later contrastive learning. Hence, we seek to reduce correlations within the graph representation bank by introducing constraints that tend to make the representations of negative samples orthogonal. Specifically, for the negative representation matrix (stacked from $\mathcal{N}$) $\mathbf{N} \in \mathbb{R}^{J \times d}$, we define its column vector $\boldsymbol{f}_l$ as the lth negative dimension vector, producing d dimension vectors $\{\boldsymbol{f}_l\}_{l=1}^{d}$. Orthogonality is usually a basic concept in dimension reduction approach including principal components analysis [12], and we also expect that latent features are independent to make sure that the redundant knowledge is minimized. We first recall the simple orthogonal constraint for negative representation matrix, i.e., $\min \|\mathbf{N}^\top \mathbf{N} - \mathbf{I}\|^2$, where $\mathbf{I} \in \mathbb{R}^{d \times d}$ is an identity matrix. However, the traditional orthogonal constraint is too strict in our cases. For example, we usually expect $\boldsymbol{f}_l^\top \boldsymbol{f}_l \gg \boldsymbol{f}_j^\top \boldsymbol{f}_l (j \neq l)$, indicating fewer correlations of the generated graph representations.

Inspired by contrastive loss, we formulate a novel bank orthogonality loss for negative representations as

$$\mathcal{L}_{BO} = -\sum_{l=1}^{d} \log \left(\frac{\exp\left(\boldsymbol{f}_l^\top \boldsymbol{f}_l / \tau\right)}{\sum_{l'=1}^{d} \exp\left(\boldsymbol{f}_l^\top \boldsymbol{f}_{l'} / \tau\right)} \right). \tag{7}$$

From Equation (7), we can observe that minimizing $\mathcal{L}_{BO}$ can result in $\boldsymbol{f}_l^\top \boldsymbol{f}_l \to 1$ and $\boldsymbol{f}_j^\top \boldsymbol{f}_l \to -1$, attempting to decorrelate negative representations.

4.4 Adversarial Learning Framework

Combining maximizing contrastive learning loss and minimizing two bank regularization losses during negative representation bank updating, we formulate our finally adversarial objective function as follows:

$$\begin{aligned} \theta^\star, \mathcal{N}^\star &= \arg\min_{\theta} \max_{\mathcal{N}} \mathcal{L}_{CL}(\theta, \mathcal{N}) + \arg\min_{\mathcal{N}} \mathcal{L}_{reg}(\mathcal{N}) \\ \mathcal{L}_{reg}(\mathcal{N}) &= \lambda_1 \mathcal{L}_{BD}(\mathcal{N}) + \lambda_2 \mathcal{L}_{BO}(\mathcal{N}), \end{aligned} \tag{8}$$

where λ_1 and λ_2 are two weights to balance the contributions of two regularization losses.

Unfortunately, we find it challenging to obtain a local minimal solution when the objective function is directly minimized as in Equation (8). To tackle this issue, motivated by the optimization technique in adversarial networks [2], we adopt a group of gradient descent and gradient ascent to optimize the parameters in the GNN-based encoder and negative representations, respectively. Formally,

$$\mathbf{n}_j \longleftarrow \mathbf{n}_j + \eta \left(\frac{\partial \mathcal{L}_{CL}(\theta, \mathcal{N})}{\partial \mathbf{n}_j} - \frac{\partial \mathcal{L}_{reg}(\theta, \mathcal{N})}{\partial \mathbf{n}_j} \right), \forall j, \tag{9}$$

$$\theta \longleftarrow \theta - \eta \frac{\partial \mathcal{L}_{CL}(\theta, \mathcal{N})}{\partial \theta}, \tag{10}$$

where η denotes the learning rate. In our framework, the negative representations are first randomly initialized with graph samples. Then we alternately update the parameters in the graph encoding branch and adversarial generation branch. As shown in Section 5.6, empirical convergence

ALGORITHM 1: Learning Algorithm of GraphACL

Input: Unlabeled data $\{\mathcal{G}_1, \ldots, \mathcal{G}_M\}$.
Parameter: Graph encoder parameter θ, momentum graph encoder parameter ϕ and negative sample bank $\mathcal{N}$=$\{\mathbf{n}_1, \ldots, \mathbf{n}_J\}$.
Output: Momentum graph encoder g_ϕ.

```
Initialize encoder parameter with a Xavier initialization and {n_1, ..., n_J} by randomly selected samples.
while not convergence do
    Sample B graphs from the training set to make up a batch;
    Produce augmented views Ĝ_i and Ĝ'_i for each graph;
    Obtain global-view graph representations z_i and z'_i through the graph encoding branch;
    Calculate loss function by Equation (8);
    Update N by gradient ascent in Equation (9);
    Update θ by gradient descent in Equation (10);
    Updating φ by momentum update in Equation (11).
end while
```

could be observed in practice, which is in line with the results of recent adversarial models [62]. Finally, we adopt momentum updating for parameters in the momentum graph encoder as

$$\phi \leftarrow \gamma\phi + (1-\gamma)\theta, \tag{11}$$

where γ is a momentum coefficient set to 0.99 following Reference [28]. Only the parameters θ are updated by back-propagation, and the parameters ϕ are evolved smoothly through momentum update. The overall learning algorithm of the GraphACL is illustrated in Algorithm 1.

Computational Complexity. As for the computational complexity, we first analyze the graph encoding branch. Given a graph, d_0 is the dimension of node attributes, $||A||_0$ is the number of nonzero elements in the adjacency matrix, K is the number of GNN layers, and $|V|$ is the number of nodes. The branch takes $O(K||A||_0 d_0 + K|V|d_0^2)$ computational time for each graph. During contrastive learning, the objective takes $O(dJN)$ computational time, where B is the batch size and M is the number of graph samples while two regularization losses take $O(Jd^2+d^2J)$. Note that both the number of negatives J and hidden dimension d are limited in our application. The complexity of GraphACL mainly depends on the graph encoding branch. Consequently, the complexity of the proposed GraphACL and the classic graph contrastive learning method GraphCL [83] is both $O(K||A||_0 d_0 + K|V|d_0^2)$ for each graph, which is linearly dependent on $|V|$ and $||A||_0$.

4.5 Model Analysis

In this section, we seek to study how our adversarial updating of negative representations can help graph contrastive learning. We look into our contrastive loss in Equation (4). First, we calculate the gradient of $\mathcal{L}_{CL}$ for $\mathbf{n}_j$:

$$\frac{\partial \mathcal{L}_{CL}}{\partial \mathbf{n}_j} = \frac{1}{M\tau} \sum_{i=1}^{M} \frac{e^{\mathbf{z}_i^\top \mathbf{n}_j/\tau} \cdot \mathbf{z}_i}{e^{\mathbf{z}_i^\top \mathbf{z}'_i/\tau} + \sum_{j'=1}^{J} e^{\mathbf{z}_i^\top \mathbf{n}_{j'}/\tau}}. \tag{12}$$

Note that the denominator is constant for each negative representation $\mathbf{n}_j$, and from the numerator, we can observe that the closer a negative example is to the positive query, the closer the example will move toward to query during updating each negative representation. In this way, the negatives gradually track the query graph and the informative negatives can be generated, thus

benefiting the optimization of the graph encoding branch in contrastive learning [65]. Further, we calculate the gradient for $\mathbf{z}_i$,

$$\frac{\partial \mathcal{L}_{CL}}{\partial \mathbf{z}_i} = -\frac{1}{M\tau}\left(\left(1 - p_{\mathbf{z}'_i}\right) \cdot \mathbf{z}'_i - \sum_{j=1}^{J} p_{\mathbf{n}_j} \cdot \mathbf{n}_j\right), \tag{13}$$

where

$$p_{\mathbf{h}} = \frac{\exp\left(\mathbf{z}_i^\top \mathbf{h}/\tau\right)}{\exp\left(\mathbf{z}_i^\top \mathbf{z}'_i/\tau\right) + \sum_{j=1}^{J} \exp\left(\mathbf{z}_i^\top \mathbf{n}_j/\tau\right)}. \tag{14}$$

Without loss of generality, assuming the first Q negative representations are identical, i.e., $\mathbf{n}_1 = \cdots = \mathbf{n}_Q$, not only the diversity of gradient space is limited but also a large accumulated gradient in the orientation of $\mathbf{n}_1$ are expected for every positive sample $\mathbf{z}_i$. This bias may make it hard to push each positive query closer to its augmentation and even result in representation collapse while the bank divergence loss can well release this issue.

Finally, we analyze the benefit of the bank orthogonality loss. We assume that the rank of $\mathbf{N}$ is r, and the negative representation matrix can be decomposed into two matrices $\mathbf{N} = \mathbf{A}\hat{\mathbf{N}}$ where $\hat{\mathbf{N}} \in \mathbb{R}^{r\times d}, \mathbf{A} \in \mathbb{R}^{J\times r}$. Since r is small, we assume that $r + 1 < d$ in some cases. Let $\hat{\mathbf{n}}_j$ denote the jth row vector of $\hat{\mathbf{N}}$. From Equation (13), the gradient of $\mathcal{L}_{CL}$ is limited in the linear subspace $span(\{\hat{\mathbf{n}}_1, \ldots, \hat{\mathbf{n}}_r, \mathbf{z}'_i\})$ with rank at most $r+1$, which seriously damages the optimization process of the graph representation learning with the above hidden constraints. From the analysis, the strong correlation of negative representations impairs contrastive learning optimization, and instead our bank orthogonality loss is designed to resolve the problem.

Feasibility Analysis. The aim of introducing negative samples is to generate uniform representations in the embedding space [68], which can avoid collapsing into trivial solutions when pulling positive pairs closer [85]. To achieve this, in previous works, negative samples from the mini-batch should be diverse, and the batch size should be large enough to ensure the uniformity of learned representations. Compared with the previous strategy, our method leverages a learnable manner to generate diverse negative samples instead. These negative samples can satisfy the requirement of divergence in graph contrastive learning. Moreover, our challenging negative samples with less correlation can benefit the optimization process more. Although our framework initializes the negatives randomly, we utilize an adversarial learning technique to update these dense representations, which maximizes the contrastive objective to provide challenging negatives. Since we have a range of positive samples in the batch, each negative sample would try to approach some of the queries in the batch, but it is difficult to have the same representation as one positive sample during adversarial learning. At the same time, by minimizing the contrastive learning objective, the positive samples would be far away from these negatives. With these alternative procedures, these negatives would remain challenging but different from these positives mostly.

4.6 Model Extension

Furthermore, we integrate the recent effective contrastive learning method BYOL [22] into our framework. Note that BYOL aims to maximize the consistency of positive pairs using an asymmetric architecture without using negative samples, which achieves superior performance for visual representation learning. To adapt it into our framework, we additionally maximize the distance of negative samples to query examples, which can make the best of challenging negative samples for enhanced graph representation learning.

In detail, we utilize an asymmetric architecture where the projector head is only added upon the graph encoder and the output representations in the calculation of contrastive learning loss

are $\mathbf{z}_i = \psi_\phi(g_\phi(\hat{\mathcal{G}}_i))$ and $\mathbf{h}'_i = g_\phi(\hat{\mathcal{G}}'_i)$. Formally, we rewrite Equation (4) as

$$\mathcal{L}_{CL} = \frac{1}{M}\sum_{i=1}^{M}\left(\left\|\frac{\mathbf{z}_i}{||\mathbf{z}_i||} - \frac{\mathbf{h}'_i}{||\mathbf{h}'_i||}\right\| - \frac{1}{J}\sum_{j=1}^{J}\left\|\frac{\mathbf{z}_i}{||\mathbf{z}_i||} - \frac{\mathbf{n}_j}{||\mathbf{n}_j||}\right\|\right), \tag{15}$$

where $||\cdot||$ calculates the L_2 norm. The asymmetric architecture of contrastive learning can empirically prevent collapsed solutions (i.e., producing similar representations for all inputs), especifically when the batch size is small [22]. We denote our extension using this contrastive loss as GraphACL*.

5 EXPERIMENTS

In this part, we conduct extensive experiments on a variety of graph classification as well as transfer learning datasets to verify the effectiveness of the proposed GraphACL. We aim to answer the following research questions:

- **RQ1:** How does the model GraphACL perform in comparison to the state-of-the-art baseline methods for unsupervised graph-level representation learning?
- **RQ2:** How do different components affect the performance of GraphACL?
- **RQ3:** What are the effects of the model hyper-parameters on the performance?
- **RQ4:** What are the effects of different encoder architectures and graph augmentation strategies on the results?
- **RQ5:** How is the convergence of the proposed model under different datasets?
- **RQ6:** How does our approach GraphACL perform under the setting of transfer learning task?

We first introduce the experimental settings, and then answer the above six research questions.

5.1 Experimental Settings

5.1.1 Evaluation Datasets. To demonstrate the advantages of our proposed GraphACL, we conduct extensive experiments on six benchmark datasets[4] following References [60, 82, 83], which are well known in the graph classification task. Specifically, we adopt three bioinformatics datasets, i.e., DD [10], PROTEINS [4], and MUTAG [41], and three social network datasets, i.e., COLLAB [81], IMDB-B, and IMDB-M. Following References [60, 82], all-ones encoding is used as node attributes when they are not directly accessible.

For bioinformatics datasets, DD [10] contains graphs of protein structures, where the nodes denote amino acids, and an edge would be built when the distance between two nodes is smaller than 6 angstroms. PROTEINS contains graphs [4] in which nodes denote secondary structure elements, and an edge would be built when two nodes are close in the one-dimensional (1D) or 3D view. MUTAG [41] is a chemical compound dataset that shows different influences on a bacterium. We aim to determine whether these protein structures are enzymes or not.

For social network datasets, IMDB-B [81] is a movie-collaboration dataset where nodes stand for actors/actresses, while an edge would be built when two nodes both show up in one movie. IMDB-M [81] is an extended version of IMDB-B and composed of ego-networks obtained from the Romance, Comedy, and Sci-Fi genres. COLLAB [81] is a scientific-collaboration dataset with three categories, condensed matter physics, high-energy physics, and astrophysics.

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

5.1.2 Compared Methods. The proposed approach GraphACL is compared with the following baselines, which can be classified into three categories, i.e., graph kernel approaches, classic unsupervised learning approaches, and graph contrastive learning approaches, as described below.

Graph kernel approaches:

— *Graphlet* [58]: This compares a pair of graphs by counting or sampling graphlets, which are often small-scale substructures or limited subgraphs.
— ***Shortest Path (SP)*** *Kernel* [3]: This is a classical graph kernel based on shortest paths, which quantifies the similarity between two graph samples according to features of the shortest paths between every node pair from two graph samples, respectively.
— ***Weisfeiler–Lehman (WL)*** *Kernel* [57]: This is a graph kernel based on the well-known Weisfeiler–Lehman test of isomorphism [72] implemented by identifying subtrees of different depths.
— ***Deep Graph Kernel (DGK)*** [81]: This is a holistic framework that learns the hidden representations of sub-structures to exploit structural information.

Traditional unsupervised learning approaches:

— *Sub2Vec* [1]: This provides an unsupervised approach that preserves graph structural information by inferring interpretable representations of decomposed subgraphs.
— *Graph2Vec* [50]: This is a well-known unsupervised method to obtain data-driven graph-level representations with structured semantics embedded.

Graph contrastive learning approaches:

— *InfoGraph* [60]: It generates discriminative graph-level representations via mutual information maximization between the graph-level representations and patch-level representations of different scales.
— *GraphCL* [83]: It introduces some basic graph augmentation techniques that accommodate different priors and then performance contrastive learning over graphs using the classic framework [7].
— *CuCo* [8]: It combines the concepts of curriculum learning and contrastive learning, which automatically choose the negative samples during optimization for unsupervised graph-level representation learning.
— *JOAO* [82]: It proposes a bi-level optimization framework to adaptively determine proper augmentation strategies for various datasets, which tackles the issue that GraphCL [83] needs prior knowledge.
— *AD-GCL* [62]: This utilizes an adversarial learning scheme to generate effective augmented views for graph samples, which allows GNNs to capture significant information during the optimization process.
— *RGCL* [45]: This incorporates invariant rationale discovery into graph contrastive learning, which utilizes a rationale generator to provide rationale-aware augmented views.

5.1.3 Parameter Settings. For our model GraphACL, we adopt GIN [79] as our GNN-based encoder following previous methods [79], which is composed of two layers of graph convolution and a sum-pooling layer. The embedding dimensions in the hidden layers are set to 512 for all datasets. The number of negative samples in the bank is set to 512, which is the same as the queue length or the number of negative samples in graph contrastive learning methods [8, 82, 83]. The two introduced weights λ_1 and λ_2 are both set to 1 as default. The model is trained via Adam optimizer [38] in which the initial learning rate is set to 0.0001.

Table 2. Compared Performance on Six Graph Classification Datasets

Methods	DD	PROTEINS	MUTAG	COLLAB	IMDB-B	IMDB-M
Graphlet (2009)	72.54 ± 3.83	71.67 ± 0.55	81.66 ± 2.11	56.30 ± 0.60	65.87 ± 0.98	43.89 ± 0.38
SP (2005)	75.47 ± 3.46	75.07 ± 0.54	85.22 ± 2.43	49.80 ± 1.20	55.60 ± 0.22	37.99 ± 0.30
WL (2011)	74.02 ± 2.28	72.92 ± 0.56	80.72 ± 3.00	69.30 ± 3.44	72.30 ± 3.44	46.95 ± 0.46
DGK (2015)	74.85 ± 0.74	73.30 ± 0.82	87.44 ± 2.72	64.66 ± 0.50	66.96 ± 0.56	44.55 ± 0.52
Sub2Vec (2018)	54.33 ± 2.44	53.03 ± 5.55	61.05 ± 15.80	55.26 ± 1.54	55.26 ± 1.54	36.67 ± 0.83
Graph2vec (2017)	70.32 ± 2.32	73.30 ± 2.05	83.15 ± 9.25	71.10 ± 0.54	71.10 ± 0.54	46.32 ± 1.44
InfoGraph (2020)	72.85 ± 1.78	74.44 ± 0.53	89.01 ± 1.13	70.65 ± 1.13	71.11 ± 0.88	49.69 ± 0.53
GraphCL (2020)	78.62 ± 0.40	74.39 ± 0.45	86.80 ± 1.34	71.36 ± 1.15	71.14 ± 0.44	48.49 ± 0.63
CuCo (2021)	76.93 ± 0.83	74.32 ± 0.31	88.74 ± 1.18	69.43 ± 0.26	70.47 ± 0.31	47.97 ± 0.12
JOAO (2021)	77.32 ± 0.54	74.55 ± 0.41	87.35 ± 1.02	69.50 ± 0.36	70.21 ± 3.08	47.22 ± 0.41
AD-GCL (2021)	74.49 ± 0.52	73.59 ± 0.65	89.25 ± 1.45	73.32 ± 0.61	71.57 ± 1.01	49.04 ± 0.53
RGCL (2022)	78.86 ± 0.48	75.03 ± 0.43	87.66 ± 1.01	70.92 ± 0.65	71.85 ± 0.84	–
GraphACL (Ours)	79.05 ± 0.51	75.29 ± 0.46	89.88 ± 1.07	74.26 ± 0.48	**74.53 ± 0.39**	**51.65 ± 0.34**
GraphACL*(Ours)	**79.34 ± 0.42**	**75.47 ± 0.38**	**90.21 ± 0.94**	**74.72 ± 0.55**	74.29 ± 0.67	51.33 ± 0.56

We report the mean and the standard deviation of prediction accuracy over five runs (in %) with different random seeds. The best results are displayed in bold.

5.1.4 Protocol. We evaluate our method using the standard evaluation protocols in References [8, 60]. In detail, the classification accuracy over 10 folds of cross-validation using LIBSVM [5] are reported. We also execute five runs with various random seeds and return both the mean accuracy (in %) and the standard deviation.

5.2 Experimental Results (RQ1)

We thoroughly compare our proposed approach GraphACL with state-of-the-art baselines and organize all the compared results in Table 2. From the results, we have the following observations:

— By comparing four graph kernel approaches, we observe that their performance has a huge variance on different datasets. For example, DGK performs best on bioinformatics datasets but performs much worse than WL on social network datasets. Perhaps the reason is that hand-crafted embeddings is different to connect with underlying graph semantics, since they cannot be learned automatically.
— Graph contrastive learning approaches typically outperform kernel methods and classic unsupervised learning approaches (i.e., Sub2vec and Graph2vec). We conclude the reason is that graph contrastive learning approaches are capable of capturing more discriminative information from graph-structured data using prior information, demonstrating the potential capability for learning effective graph-level representations.
— Our framework GraphACL obtains the best performance consistently against baselines on all datasets, which validates the effectiveness of the proposed GraphACL. In comparison to the state-of-the-art method RGCL, the improvement of performance on MUTAG and COLLAB is 2.53% and 4.71%, respectively. Considering that the graph dataset has a very huge variance for different types, our improvement is very significant.
— Our extension GraphACL* obtains a slight performance gain compared with GraphACL on most datasets. This observation may result from that the asymmetric architecture empirically benefits the optimization of contrastive learning [22]. Therefore, we utilize GraphACL* as the default in the following analysis.

Discussion on the Remarkable Improvement. Although graph contrastive learning has been widely explored in previous works, negative mining strategies are under-explored for learning

Table 3. Ablation Study of Different Model Variants (in %) on Six Datasets

Methods	DD	PROTEINS	MUTAG	COLLAB	IMDB-B	IMDB-M
Variant-1	77.84 ± 0.71	73.85 ± 0.54	88.86 ± 1.23	72.69 ± 0.61	72.35 ± 0.82	50.23 ± 0.45
Variant-2	78.63 ± 0.68	74.26 ± 0.36	89.65 ± 0.98	74.13 ± 0.42	73.42 ± 0.93	50.79 ± 0.51
Variant-3	78.38 ± 0.63	74.94 ± 0.43	89.39 ± 1.04	73.61 ± 0.68	73.57 ± 0.78	51.08 ± 0.37
Variant-4	77.26 ± 0.89	73.51 ± 0.62	88.30 ± 1.15	71.56 ± 0.84	71.40 ± 0.64	49.81 ± 0.69
GraphACL+	78.94 ± 0.58	75.13 ± 0.33	90.08 ± 0.89	74.58 ± 0.65	74.22 ± 0.59	**51.42 ± 0.52**
GraphACL* (Full Model)	**79.34 ± 0.42**	**75.47 ± 0.38**	**90.21 ± 0.94**	**74.72 ± 0.55**	**74.29 ± 0.67**	51.33 ± 0.56

discriminative graph-level representations. Obviously, uninformative and low-quality negatives will hugely misguide the optimization of contrastive objective, while our GraphACL introduces two key components to tackle this issue: (i) the introduction of an adversarial generation branch (the adversarial generation branch is trained in an adversarial manner, which is capable of producing informative negative samples to enhance contrastive learning) and (ii) the introduction of two bank regularization losses. The two losses aim to increase the diversity and decrease the correlation of learned representations, alleviating the ill-conditioned issue that may occur in the optimization process.

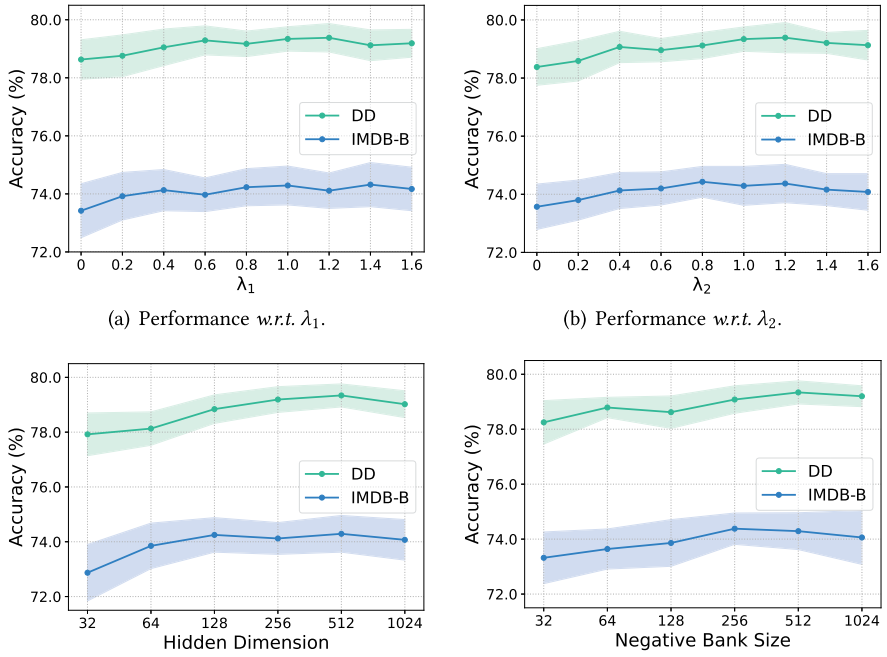
5.3 Ablation Study (RQ2)

To figure out the key role of each component in our model, we perform ablation studies to study their contribution. Particularly, we design the following model variants:

— Variant-1: We do not produce both bank divergence loss and bank orthogonality loss (i.e., $\lambda_1 = \lambda_2 = 0$), and the model is trained only through adversarial learning.
— Variant-2: The bank divergence loss is removed (i.e., $\lambda_1 = 0$).
— Variant-3: The bank orthogonality loss is removed (i.e., $\lambda_2 = 0$).
— Variant-4: The gradient ascent of graph contrastive loss in Equation (9) is removed and the adversarial generation branch is trained with only two bank regularization losses.
— GraphACL+: The generated negative samples are combined with other negative samples in the current minibatch to serve as negatives.

The experimental results are recorded in Table 3, and we can observe that different datasets have a similar observation. We summarize the following findings:

— First, from the comparison between the full model with the first three model variants, we can see significant performance increasement after these corresponding modules are combined, which illustrates the significance of every module in the model. They contribute greatly to the final performance of the model.
— Second, when we remove either (i.e., Variant-2 and Variant-3) or both (i.e., Variant-1) bank regularization losses, the performance of the model deteriorates significantly, where Variant-1 has the worst results. This fully demonstrates that both the bank divergence loss and the bank orthogonality loss are beneficial to model performance, because they promise the high divergence and low correlation of the negative samples, respectively.
— Third, Variant-4 shows much worse performance compared with our full model, which hence illustrates the importance and benefit of adversarial learning in our adversarial generation branch. The reason is that our negative samples can quickly adaptively track the change of the positive sample through gradient updating in the process of adversarial learning, and instead adversarial learning can assist the framework in generating informative negative samples for more rigorously training the graph-level representations.



(a) Performance *w.r.t.* λ_1.

(b) Performance *w.r.t.* λ_2.

(c) Performance *w.r.t.* the hidden dimension.

(d) Performance *w.r.t.* the negative bank size.

Fig. 3. Performance w.r.t. balance weights λ_1 (a), λ_2 (b), hidden dimension (c), and negative bank size (d).

— Last, from the comparison between GraphACL+ and our full model, we can observe that adding more samples cannot improve the performance evidently, indicating that the generated negative samples have enough capacity for graph contrastive learning. Therefore, to improve the efficiency, we would not utilize other negative samples in the current minibatch.

5.4 Parameter Sensitivity (RQ3)

In this subsection, we study the parameter sensitivity of the proposed GraphACL. In particular, we report the influence of different balance weights and embedding dimensions in the hidden layer on two datasets DD and IMDB-B.

Effect of Different Balance Weights. We analyze the effect of λ_1 and λ_2 while fixing all other parameters. As shown in Figures 3(a) and 3(b), we first fix λ_2 to 1 and evaluate the performance by varying λ_1 from 0 to 2 and varying λ_2 with λ_1 fixed to 1. For both balance weights, the accuracy first increases and then keeps at a relatively high level. The result is not sensitive to both balance weights in the range of $[0.5, 1.5]$. For our GraphACL, λ_1 and λ_2 are both set to 1 as a default.

Effect of Different Hidden Dimensions. Additionally, we analyze the influence of different embedding dimensions in the hidden layers d. We vary d in $\{32, 64, 128, 256, 512, 1024\}$ while fixing all other parameters to the ones yielding the best performance. We show the results in Figure 3(c). It can be observed that when the hidden dimension is small, the performance gradually improves as the hidden dimension increases. When the hidden dimension reaches 256 or 512, the model achieves the best results. However, too-large dimensions will rapidly expand our parameter space, which could lead to over-fitting and poor generalization, resulting in sub-optimal performance.

Effect of Different Bank Sizes. Last, we study the impact of different bank sizes by varying the size in $\{32, 64, 128, 256, 512, 1024\}$. The results are shown in Figure 3(d). The results demonstrate

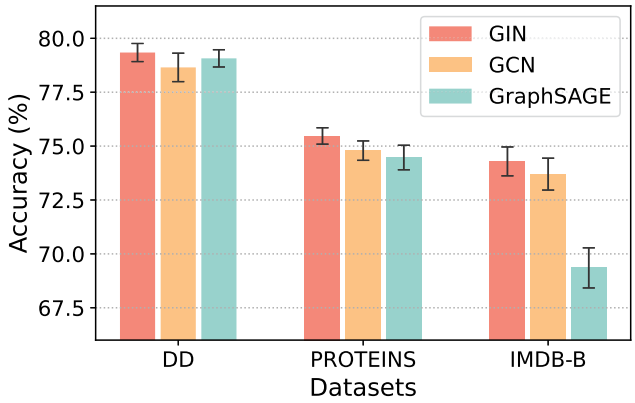


Fig. 4. Performance with different encoder architectures.

that model performance is enhanced as bank size increases but only up to a certain point. This improvement is primarily due to the fact that contrastive learning requires a sufficient quantity of negative samples. However, once the bank size exceeds 512, the enhancement in model performance starts to plateau. Balancing performance improvement with computational efficiency, we set the default bank size to 512.

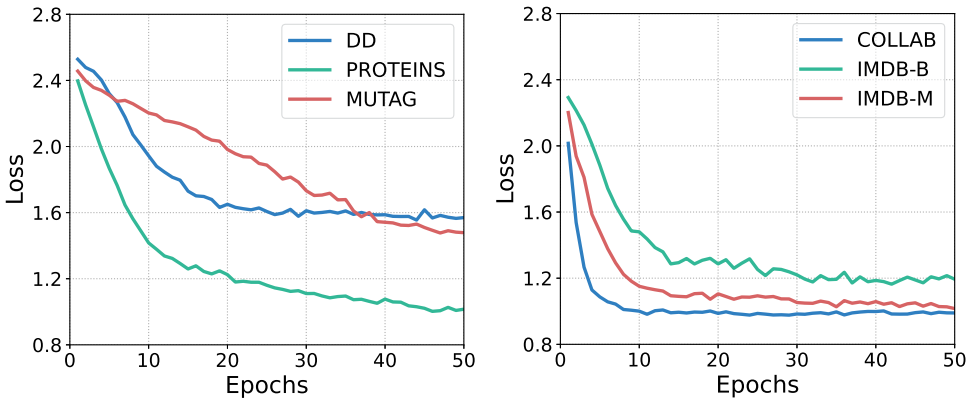
5.5 Impact of Different Encoder (RQ4)

In this part, we investigate the impact of different GNN-based encoder architectures. In particular, we select three popular types of GNNs (i.e., GCN [40], GraphSAGE [24], and GIN [79]) and evaluate their performance in our GraphACL on three datasets, i.e., DD, PROTEINS, and IMDB-B. The result is shown in Figure 4. It can be found that GIN slightly performs better than the other two encoders on three datasets, demonstrating the high model capacity of GIN. This also explains the reason for choosing GIN as the basic model for all graph contrastive learning methods. Moreover, we can observe that GraphSAGE achieves much worse performance on IMDB-B compared with the other two encoders, which validates the importance of choosing a proper GNN-based encoder in the first branch.

5.6 Empirical Convergence (RQ5)

As shown in Figure 5, we illustrate the training loss curves of our GraphACL on six datasets, which can be divided into two categories: bioinformatics datasets and social network datasets. We can observe that iterative gradient updates to the graph encoding branch and the adversarial generation branch on all datasets work well and achieve convergence empirically, which is a common case in a range of adversarial approaches [9, 21, 66]. This phenomenon further validates the feasibility of alternative optimization for adversarial learning.

Additionally, we can see that most datasets can converge within 20 epochs, which also validates the accelerated convergence speed when optimizing our GraphACL. The reason is that our negative samples can quickly adaptively track the change of the positive sample through gradient updating, so that the learned graph-level representations can always maintain a high level of discrimination ability. Moreover, the two regularized losses proposed in our framework can better guide the representations of negative samples to be better dispersed around the distribution of positive sample data and provide a piece of good gradient information for rapid updating of the model.



(a) Bioinformatics (b) Social Networks

Fig. 5. The training curves of the GraphACL on two groups of datasets.

5.7 Transfer Learning (RQ6)

Here we perform transfer learning on molecular chemical property prediction tasks following Reference [32], which pre-trains and finetunes different approaches on different large-scale **Open Graph Benchmark (OGB)** [31] datasets under the different pre-training schemes for transferability evaluation.

Experiments Setting. In this experiment, we evaluate the effectiveness of our model on eight OGB [31] molecule property prediction datasets. When pre-training, following Reference [79], we leverage the same GNN-based encoder (GIN) with 300-dimensional hidden units and a mean-pooling summarization operation for performance comparisons. As for fine-tuning on a downstream task, a classifier head is fully optimized on top of the pre-trained GNN. We perform 10-fold cross-validation and report the mean together with the standard deviation of ROC-AUC scores over five runs. For baseline methods, GraphACL is compared with non-pretrain (direct supervised learning) and various pre-training strategies. Besides four graph contrastive learning methods, GraphCL [83], JOAO [82], AD-GCL [62], and RGCL [45], we further include five extra pre-training techniques, i.e., EdgePred [39], Infomax [64], AttrMasking [32], ContextPred [32], and GraphPartition [84].

Pre-training dataset. We leverage a part of the ZINC15 database [18, 49] with 200,000,000 unlabeled molecules for self-supervised graph pre-training. The dataset is the same datasets as in Reference [32] for a fair comparison.

Downstream task datasets. Eight larger binary graph classification datasets contained in Moleculenet [74] serving as downstream tasks are used to evaluate model performance, where the scaffold split scheme [6] is used for dataset split.

Performance Analysis. The compared performance of different approaches is recorded in Table 4. Among all competing baselines, our method (GraphACL and GraphACL*) achieves state-of-the-art performance results on six of eight datasets. It is worth noting that our GraphACL* gains a 11.3% performance enhancement on average compared with the non-pretrain baseline, which well indicates the superiority of GraphACL on transfer learning. In this task, the best results on each dataset are achieved by different baseline methods. Because the characteristics of different downstream tasks vary greatly, it is difficult to have a unified framework that can comprehensively capture the common knowledge of different datasets. However, our approach achieves optimal

Table 4. Results on Downstream Molecular Property Prediction Benchmarks

Methods	BBBP	BACE	ClinTox	HIV	MUV	SIDER	Tox21	ToxCast	AVG.
No Pre-Train	65.8 ± 4.5	70.1 ± 5.4	58.0 ± 4.4	75.3 ± 1.9	71.8 ± 2.5	57.3 ± 1.6	74.0 ± 0.8	63.4 ± 0.6	67.0
EdgePred (2016)	67.3 ± 2.4	79.9 ± 0.9	64.1 ± 3.7	76.3 ± 1.0	74.1 ± 2.1	60.4 ± 0.7	76.0 ± 0.6	64.1 ± 0.6	70.3
Infomax (2019)	68.8 ± 0.8	75.9 ± 1.6	69.9 ± 3.0	76.0 ± 0.7	75.3 ± 2.5	58.4 ± 0.8	75.3 ± 0.5	62.7 ± 0.4	70.3
AttrMasking (2020)	64.3 ± 2.8	79.3 ± 1.6	71.8 ± 4.1	77.2 ± 1.1	74.7 ± 1.4	61.0 ± 0.7	76.7 ± 0.4	**64.2 ± 0.5**	71.7
ContextPred (2020)	68.0 ± 2.0	79.6 ± 1.2	65.9 ± 3.8	77.3 ± 1.0	75.8 ± 1.7	60.9 ± 0.6	75.7 ± 0.7	63.9 ± 0.6	70.9
GraphPartition (2020)	70.3 ± 0.7	79.6 ± 1.8	64.2 ± 0.5	77.1 ± 0.7	75.4 ± 1.7	61.0 ± 0.8	75.2 ± 0.4	63.2 ± 0.3	70.7
GraphCL (2020)	69.7 ± 0.7	75.4 ± 1.4	76.0 ± 2.7	78.5 ± 1.2	69.8 ± 2.7	60.5 ± 0.9	73.9 ± 0.7	62.4 ± 0.6	70.8
JOAO (2021)	70.2 ± 1.0	77.3 ± 0.5	81.3 ± 2.5	76.7 ± 1.2	71.7 ± 1.4	60.0 ± 0.8	75.0 ± 0.3	63.0 ± 0.5	71.9
AD-GCL (2021)	70.0 ± 1.1	78.5 ± 0.8	79.8 ± 3.5	78.3 ± 1.0	72.3 ± 1.6	**63.3 ± 0.8**	76.5 ± 0.8	63.1 ± 0.7	72.7
RGCL (2022)	71.4 ± 0.7	76.0 ± 0.8	83.4 ± 0.9	77.9 ± 0.8	76.7 ± 1.0	61.4 ± 0.6	75.2 ± 0.3	63.3 ± 0.2	73.2
GraphACL (Ours)	72.5 ± 0.9	**80.4 ± 0.7**	83.7 ± 1.0	78.4 ± 0.9	76.1 ± 1.5	62.3 ± 0.8	**76.8 ± 0.3**	63.8 ± 0.5	74.3
GraphACL*(Ours)	**73.3 ± 0.5**	80.1 ± 1.2	**85.0 ± 1.6**	**78.9 ± 0.7**	**76.9 ± 1.2**	62.6 ± 0.6	76.2 ± 0.6	64.1 ± 0.4	**74.6**

We show the mean along with the standard deviation of ROC-AUC scores over five runs using 10-fold cross-validation. The transferability of the proposed method GraphACL is better than all the baseline methods in most cases.

performance over most datasets in a consistent way. Further, compared to our state-of-the-art baseline RGCL [45] in terms of average ROC-AUC, our method yielded better consistent results on all datasets, and this comparison strongly demonstrates the effectiveness of the introduced framework GraphACL.

6 CONCLUSION

In this article, we study unsupervised graph-level representation learning, which is a long-standing problem of learning the representations of a whole graph without human supervision, and a new approach named GraphACL is proposed. GraphACL is a principled framework that consists of a graph encoding branch that generates the representations of positive samples and an adversarial generation branch that produces a bank of negative samples. The two branches are trained alternately by minimizing the contrastive loss during encoding updating while maximizing the contrastive loss over the negative samples in an adversarial manner. Moreover, two regularization losses named bank divergence loss and bank orthogonality loss have been proposed to guide the training of negative samples, which can further improve the discriminability of derived graph-level representations.

Comprehensive experiments on various publicly available graph classification datasets demonstrate the effectiveness of the proposed framework over existing state-of-the-art baselines. In future works, we would extend GraphACL to a broader range of applications, including materials science and recommender systems, and further explore more effective graph representation learning without resorting to negative samples, alleviating the cost in terms of time and memory.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B. Aditya Prakash. 2018. Sub2vec: Feature learning for subgraphs. In *PAKDD*.

[2] Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. *ICLR*.

[3] Karsten M. Borgwardt and Hans-Peter Kriegel. 2005. Shortest-path kernels on graphs. In *ICDM*.

[4] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21, S1 (2005), i47–i56.

[5] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* 2, 3 (2011), 1–27.
[6] Bin Chen, Robert P. Sheridan, Viktor Hornak, and Johannes H. Voigt. 2012. Comparison of random forest and pipeline pilot naive bayes in prospective QSAR predictions. *J. Chem. Inf. Model.* 52, 3 (2012), 792–803.
[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*.
[8] Guanyi Chu, Xiao Wang, Chuan Shi, and Xunqiang Jiang. 2021. CuCo: Graph representation with curriculum contrastive learning. In *IJCAI*.
[9] Ming Ding, Jie Tang, and Jie Zhang. 2018. Semi-supervised learning on graphs with generative adversarial nets. In *CIKM*.
[10] Paul D. Dobson and Andrew J. Doig. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *J. Molec. Biol.* 330, 4 (2003), 771–783.
[11] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. 2014. Discriminative unsupervised feature learning with convolutional neural networks. In *NeurIPS*.
[12] George H. Dunteman. 1989. *Principal Components Analysis*. Sage.
[13] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*.
[14] Barakeel Fanseu Kamhoua, Lin Zhang, Kaili Ma, James Cheng, Bo Li, and Bo Han. 2023. Grace: A general graph convolution framework for attributed graph clustering. *ACM Trans. Knowl. Discov. Data* 17, 3 (2023), 1–31.
[15] Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. 2019. Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering* 33, 6 (2019), 2493–2504.
[16] Jiarui Feng, Yixin Chen, Fuhai Li, Anindya Sarkar, and Muhan Zhang. 2022. How powerful are k-hop message passing graph neural networks. In *NeurIPS*, Vol. 35. 4776–4790.
[17] Kimon Fountoulakis, Amit Levi, Shenghao Yang, Aseem Baranwal, and Aukosh Jagannath. 2023. Graph attention retrospective. *J. Mach. Learn. Res.* 24, 246 (2023), 1–52.
[18] Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. 2012. ChEMBL: A large-scale bioactivity database for drug discovery. *Nucleic Acids Res.* 40, D1 (2012), D1100–D1107.
[19] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*.
[20] Maoguo Gong, Hui Zhou, AK Qin, Wenfeng Liu, and Zhongying Zhao. 2022. Self-paced co-training of graph neural networks for semi-supervised node classification. *IEEE Trans. Neural Netw. Learn. Syst.* (2022).
[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NeurIPS*.
[22] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. In *NeurIPS*.
[23] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *CVPR*.
[24] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.
[25] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.
[26] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. 2022. G-mixup: Graph data augmentation for graph classification. In *ICML*. 8230–8248.
[27] Zhongkai Hao, Chengqiang Lu, Zhenya Huang, Hao Wang, Zheyuan Hu, Qi Liu, Enhong Chen, and Cheekong Lee. 2020. ASGN: An active semi-supervised graph neural network for molecular property prediction. In *KDD*.
[28] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*.
[29] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. Regal: Representation learning-based graph alignment. In *CIKM*.
[30] Qianjiang Hu, Xiao Wang, Wei Hu, and Guo-Jun Qi. 2021. Adco: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. *CVPR*, 1074–1083.
[31] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *NeurIPS*, 33 (2020), 22118–22133.
[32] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for pre-training graph neural networks. In *ICLR*.

[33] Wei Ju, Zheng Fang, Yiyang Gu, Zequn Liu, Qingqing Long, Ziyue Qiao, Yifang Qin, Jianhao Shen, Fang Sun, Zhiping Xiao, et al. 2023. A comprehensive survey on deep graph representation learning. arXiv:2304.05055. Retrieved from https://arxiv.org/abs/2304.05055

[34] Wei Ju, Yiyang Gu, Xiao Luo, Yifan Wang, Haochen Yuan, Huasong Zhong, and Ming Zhang. 2023. Unsupervised graph-level representation learning with hierarchical contrasts. *Neural Netw.* 158 (2023), 359–368.

[35] Wei Ju, Xiao Luo, Meng Qu, Yifan Wang, Chong Chen, Minghua Deng, Xian-Sheng Hua, and Ming Zhang. 2023. TGNN: A joint semi-supervised framework for graph-level classification. arXiv:2304.11688. Retrieved from https://arxiv.org/abs/2304.11688

[36] Wei Ju, Yifang Qin, Ziyue Qiao, Xiao Luo, Yifan Wang, Yanjie Fu, and Ming Zhang. 2022. Kernel-based substructure exploration for next POI recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'22)*. IEEE, 221–230.

[37] Dae Ha Kim and Byung Cheol Song. 2021. Contrastive adversarial learning for person independent facial emotion recognition. In *AAAI*.

[38] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

[39] Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. In *NeurIPS Workshop on Bayesian Deep Learning*.

[40] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[41] Nils Kriege and Petra Mutzel. 2012. Subgraph matching kernels for attributed graphs. In *ICML*.

[42] Shiyong Lan, Yitong Ma, Weikang Huang, Wenwu Wang, Hongyu Yang, and Pyang Li. 2022. Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In *ICML*. PMLR, 11906–11917.

[43] Fuxian Li, Jie Feng, Huan Yan, Guangyin Jin, Fan Yang, Funing Sun, Depeng Jin, and Yong Li. 2023. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. *ACM Trans. Knowl. Discov. Data* 17, 1 (2023), 1–21.

[44] Qian Li, Xiangmeng Wang, Zhichao Wang, and Guandong Xu. 2023. Be causal: De-biasing social network confounding in recommendation. *ACM Trans. Knowl. Discov. Data* 17, 1 (2023), 1–23.

[45] Sihang Li, Xiang Wang, An Zhang, Yingxin Wu, Xiangnan He, and Tat-Seng Chua. 2022. Let invariant rationale discovery inspire graph contrastive learning. In *ICML*.

[46] Xiao Luo, Wei Ju, Meng Qu, Yiyang Gu, Chong Chen, Minghua Deng, Xian-Sheng Hua, and Ming Zhang. 2022. Clear: Cluster-enhanced contrast for self-supervised graph representation learning. *IEEE Trans. Neural Netw. Learn. Syst.* (2022).

[47] Xiao Luo, Yusheng Zhao, Yifang Qin, Wei Ju, and Ming Zhang. 2023. Towards semi-supervised universal graph classification. *IEEE Trans. Knowl. Data Eng.* (2023).

[48] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *ICLR*.

[49] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. 2018. Large-scale comparison of machine learning methods for drug target prediction on ChEMBL. *Chem. Sci.* 9, 24 (2018), 5441–5451.

[50] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning distributed representations of graphs. arXiv:1707.05005. Retrieved from https://arxiv.org/abs/1707.05005

[51] Giannis Nikolentzos and Michalis Vazirgiannis. 2020. Random walk graph neural networks. In *NeurIPS*.

[52] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. arXiv:1807.03748. Retrieved from https://arxiv.org/abs/1807.03748

[53] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. 2021. Bag of tricks for adversarial training. *ICLR*.

[54] Yifang Qin, Hongjun Wu, Wei Ju, Xiao Luo, and Ming Zhang. 2023. A diffusion model for POI recommendation. arXiv:2304.07041. Retrieved from https://arxiv.org/abs/2304.07041

[55] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *KDD*.

[56] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. 2018. Mining point cloud local structures by kernel correlation and graph pooling. In *CVPR*.

[57] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.* 12, 9 (2011), 2539–2561.

[58] Nino Shervashidze, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. 2009. Efficient graphlet kernels for large graph comparison. In *AISTATS*.

[59] Xiangbo Shu, Binqian Xu, Liyan Zhang, and Jinhui Tang. 2022. Multi-granularity anchor-contrastive representation learning for semi-supervised skeleton-based action recognition. *IEEE Trans. Pattern Analy. Mach. Intell.* (2022).

[60] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2020. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*.
[61] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Yuanxing Ning, Philip S Yu, and Lifang He. 2021. SUGAR: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. In *WWW*.
[62] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. 2021. Adversarial graph augmentation to improve graph contrastive learning. In *NeurIPS*.
[63] Qiaoyu Tan, Xin Zhang, Ninghao Liu, Daochen Zha, Li Li, Rui Chen, Soo-Hyun Choi, and Xia Hu. 2023. Bring your own view: Graph neural networks for link prediction with personalized subgraph selection. In *WSDM*. 625–633.
[64] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep graph infomax. In *ICLR*.
[65] Feng Wang and Huaping Liu. 2021. Understanding the behaviour of contrastive loss. In *CVPR*.
[66] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. Graphgan: Graph representation learning with generative adversarial nets. In *AAAI*.
[67] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. 2019. Graph attention convolution for point cloud semantic segmentation. In *CVPR*.
[68] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*.
[69] Weilun Wang, Wengang Zhou, Jianmin Bao, Dong Chen, and Houqiang Li. 2021. Instance-wise hard negative example generation for contrastive learning in unpaired image-to-image translation. In *ICCV*.
[70] Xuemei Wei, Yezheng Liu, Jianshan Sun, Yuanchun Jiang, Qifeng Tang, and Kun Yuan. 2023. Dual subgraph-based graph neural network for friendship prediction in location-based social networks. *ACM Trans. Knowl. Discov. Data* 17, 3 (2023), 1–28.
[71] Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and Tat-Seng Chua. 2021. Contrastive learning for cold-start recommendation. In *ACMMM*.
[72] Boris Weisfeiler and Andrei A. Lehman. 1968. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Techn. Inf.* 2, 9 (1968), 12–16.
[73] Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao, and Xiaojie Guo. 2022. Graph neural networks: Foundation, frontiers and applications. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4840–4841.
[74] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: A benchmark for molecular machine learning. *Chem. Sci.* 9, 2 (2018), 513–530.
[75] Binqian Xu and Xiangbo Shu. 2023. Pyramid self-attention polymerization learning for semi-supervised skeleton-based action recognition. arXiv:2302.02327. Retrieved from https://arxiv.org/abs/2302.02327
[76] Binqian Xu, Xiangbo Shu, and Yan Song. 2022. X-invariant contrastive augmentation and representation learning for semi-supervised skeleton-based action recognition. *IEEE Trans. Image Process.* 31 (2022), 3852–3867.
[77] Binqian Xu, Xiangbo Shu, Rui Yan, Guo-Sen Xie, Yixiao Ge, and Mike Zheng Shou. 2023. Attack is good augmentation: Towards skeleton-contrastive representation learning. arXiv:2304.04023. Retrieved from https://arxiv.org/abs/2304.04023
[78] Binqian Xu, Xiangbo Shu, Jiachao Zhang, Guangzhao Dai, and Yan Song. 2023. Spatiotemporal decouple-and-squeeze contrastive learning for semisupervised skeleton-based action recognition. *IEEE Trans. Neural Netw. Learn. Syst.* (2023).
[79] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *ICLR*.
[80] Biwei Yan, Guijuan Wang, Jiguo Yu, Xiaozheng Jin, and Hongliang Zhang. 2021. Spatial-temporal chebyshev graph neural network for traffic flow prediction in iot-based its. *IEEE IoT J.* (2021).
[81] Pinar Yanardag and S. V. N. Vishwanathan. 2015. Deep graph kernels. In *KDD*.
[82] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph contrastive learning automated. In *ICML*.
[83] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. In *NeurIPS* (2020).
[84] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. 2020. When does self-supervision help graph convolutional networks? In *ICML*.
[85] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. 2021. Barlow twins: Self-supervised learning via redundancy reduction. *ICML*, 12310–12320.
[86] Jiaqi Zeng and Pengtao Xie. 2021. Contrastive self-supervised learning for graph classification. In *AAAI*.
[87] Tong Zhang, Yun Wang, Zhen Cui, Chuanwei Zhou, Baoliang Cui, Haikuan Huang, and Jian Yang. 2021. Deep wasserstein graph discriminant learning for graph classification. In *AAAI*.
[88] Yanfu Zhang, Shangqian Gao, Jian Pei, and Heng Huang. 2022. Improving social network embedding via new second-order continuous graph neural networks. In *KDD*. 2515–2523.

[89] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S. Yu. 2022. Graph neural networks for graphs with heterophily: A survey. arXiv:2202.07082. Retrieved from https://arxiv.org/abs/2202.07082

[90] Jiabo Zhuang, Shunmei Meng, Jing Zhang, and Victor S. Sheng. 2023. Contrastive learning based graph convolution network for social recommendation. *ACM Trans. Knowl. Discov. Data* 17, 8 (2023), 1–21.

[91] Wei Zhuo and Guang Tan. 2022. Efficient graph similarity computation with alignment regularization. In *NeurIPS*, Vol. 35. 30181–30193.

Received 7 April 2023; revised 2 September 2023; accepted 6 September 2023