Graph Contrastive Learning with Reinforcement Augmentation

Ziyang Liu , Chaokun Wang , Cheng Wu
School of Software, BNRist, Tsinghua University, Beijing, China
{liu-zy21, wuc22}@mails.tsinghua.edu.cn, chaokun@tsinghua.edu.cn

Abstract

Graph contrastive learning (GCL), designing contrastive objectives to learn embeddings from augmented graphs, has become a prevailing method for extracting embeddings from graphs in an unsupervised manner. As an important procedure in GCL, graph data augmentation (GDA) directly affects the model performance on downstream tasks. Currently, the GCL methods typically treat GDA as independent events, neglecting its continuity. In this paper, we regard the GDA in GCL as a Markov decision process and propose a novel graph reinforcement augmentation framework for GCL. Based on this framework, we design a Graph Advantage Actor-Critic (GA2C) model. We conduct extensive experiments to evaluate GA2C on unsupervised learning, transfer learning, and semi-supervised learning. The experimental results demonstrate the performance superiority of GA2C over the state-of-the-art GCL models. Furthermore, we verify that GA2C is more efficient than the other GCL methods with learnable GDA and provide two examples of chemical molecular graphs from ZINC-2M to demonstrate that GA2C generates meaningful augmented views, where the edge weights reflect the importance of chemical bonds in the molecule.

1 Introduction

Graph representation learning aims to extract low-dimensional representation vectors from the graph data by supervised or unsupervised learning [Cui *et al.*, 2019; Yang *et al.*, 2023; Liu *et al.*, 2022]. These representation vectors encode the abundant structural and semantic information of the graph. Graph representation learning has become an effective technique of graph mining and is applied to multiple fields including bioinformatics [Ang *et al.*, 2021; You *et al.*, 2020], social networks [Wang *et al.*, 2018; Wu *et al.*, 2023], and recommender systems [Yu *et al.*, 2022; Xu *et al.*, 2023]. Due to the sparsity of labeling information in the real world, learning representations from graphs in a self-supervised manner has become particularly crucial. Furthermore, as an important member of self-supervised representation learning on graphs, graph contrastive learning (GCL) has achieved superior performance on a series of downstream tasks such as graph classification [Suresh *et al.*, 2021; Yin *et al.*, 2022; Liu *et al.*, 2023].

As a necessary procedure in GCL, graph data augmentation (GDA) generally defines augmented views based on the original graph and directly affects the model performance on downstream tasks. The common GDA strategies include edge removing, edge perturbation, attribute masking, and so on. Prior work designs three types of GDA in total: trial-and-error methods, precomputed methods, and adversarial methods. In the trial-and-error methods [Zhu *et al.*, 2020; Thakoor *et al.*, 2021] and precomputed methods [Zhu *et al.*, 2021], GDA is frozen in the whole training. While in the adversarial methods [Suresh *et al.*, 2021; You *et al.*, 2022], GDA is learnable in the training and its parameters are updated by a pre-designed view learner. In addition, some GCL models leverage augmentation-free methods, such as encoder perturbation [Xia *et al.*, 2022], to construct self-supervised signals instead of GDA.

Motivation 1: Evolvability of augmented views. Currently, GCL with learnable augmentation has unleashed the potential of contrastive learning and achieved state-of-the-art performance on downstream tasks [Suresh *et al.*, 2021; Yin *et al.*, 2022]. Then a new potential question arises: *How does a good augmented view evolve to promote the performance of GCL?* We argue that a good augmented view should maintain the characteristics of progressive evolution, akin to the step-by-step learning progression in human cognition, where it is easier to comprehend and accept new information when there is a connection between what is learned on successive days. It corresponds to a Markov decision process, i.e., the augmented view of the current epoch is only affected by that of the last epoch.

Motivation 2: Preservation of original graph structure information. In specific graph structures, GDA strategies like edge removing or node deletion can disrupt the fundamental structure information of the original graph. For instance, in molecular property prediction, removing atoms or chemical bonds destroys the basic structure of the chemical molecule, or even alters the types of functional groups [Xia *et al.*, 2022; Wang *et al.*, 2022a]. Some existing GCL models use augmentation-free methods like encoder perturbation to address this issue and have achieved commendable results. Con-

sidering the difficulty in determining the appropriate encoder perturbation ratio (e.g., inappropriate perturbations can lead to performance degradation [Xia *et al.*, 2022]), we cannot help but wonder: *Is there a scheme that preserves the fundamental structure information of the original graph without perturbing the encoder?*

In this study, we propose the reinforced GDA where the view learner adjusts GDA parameters actively based on the previous state (Motivation 1). Our view learner in GDA is regarded as an agent that interacts with the encoder network. Moreover, the view learner learns the weights of edges in the graph, without deleting any node, edge, or attribute in the graph. As a result, the augmented view preserves the original graph structure information completely (Motivation 2). Applying reinforcement learning to GDA is non-trivial because of two primary challenges: Firstly, the existing reinforcement learning methods on graphs are not adequately suited for GDA [Ju *et al.*, 2023; Munikoti *et al.*, 2023]; secondly, designing an effective reward function for graph-based environments remains unclear. To solve the two challenges, we propose a novel graph reinforcement augmentation (GRA) framework for GCL and design a Graph Advantage Actor-Critic (GA2C) model based on this framework. In GA2C, the joint use of the Actor submodel and Critic submodel contributes to the progressive evolution of the augmented view. Also, the reward function in GA2C is designed as the negative mutual information between two augmented graph embeddings to reduce the redundant information between two views.

The main contributions of this work are as follows:

- We propose the GRA framework, a novel type of GDA for GCL, based on reinforcement learning. This framework formulates a Markov decision process for GDA and preserves the original graph structure information (Section 4.1).
- We design the GA2C model, as an instantiation of the GRA framework, to achieve continuous and learnable graph data augmentation (Section 4.2).
- On 13 public datasets, we demonstrate that GA2C outperforms the state-of-the-art GCL models in unsupervised, transfer, and semi-supervised learning. (Section 5).

2 Related Work

Graph contrastive learning with frozen GDA parameters. Traditional GCL models like GraphCL [You *et al.*, 2020] and InfoGraph [Sun *et al.*, 2020] adopt frozen GDA parameters to generate augmented views. Generally, these models set GDA parameters empirically by trial and error. Different from the trial-and-error methods, GCA [Zhu *et al.*, 2021] adopts precomputed methods, i.e., it calculates GDA parameters based on some indicators of centrality in the graph (e.g., degree centrality) and uses these precomputed parameters to augment the graph. However, there is a risk in frozen GDA parameters that the redundant information is captured by the InfoMax principle [Tschannen *et al.*, 2020]. The redundant information hinders the optimal performance of GZL on downstream tasks.

Graph contrastive learning with learnable augmentation. To reduce the negative impact of redundant information, learnable augmentation is introduced into GCL, i.e., the parameters

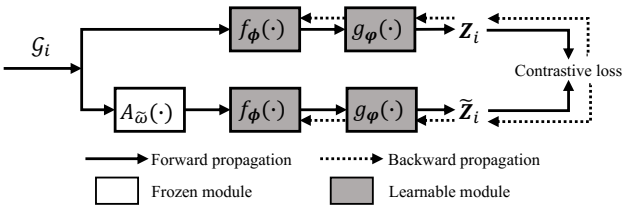

Figure 1: Framework of GCL with frozen GDA parameters.

of GDA are learned automatically in the training. For example, AD-GCL [Suresh *et al.*, 2021] adopts adversarial training, i.e., the contrastive optimization aims to i) maximize the correspondence between the embeddings of different views when fixing GDA and updating network encoding; ii) minimize the correspondence between the embeddings of different views when fixing network encoding and updating GDA. Also, inspired by image manifolds, the study in [You *et al.*, 2022] extends the frozen GDA parameters to the learnable ones, and leverages both principles of information minimization and information bottleneck to regularize the learned GDA parameters. JOAO [You *et al.*, 2021] is an automated augmentation selection framework for GraphCL that aligns with best practices. As an enhanced version of JOAO, JOAOv2 [You *et al.*, 2021] utilizes an augmentation-aware projection head to counter training distribution distortions caused by aggressive augmentations. The key difference between the above methods and ours is that our method captures the evolutionary aspect of augmented views and preserves the original graph structure information.

Augmentation-free graph contrastive learning. In the literature, augmentation-free GCL models learn embeddings from graphs without using GDA. These models typically leverage encoder perturbation or embedding perturbation instead of GDA to construct contrast pairs. For example, SimGRACE [Xia *et al.*, 2022] takes the raw graph as the input and uses the graph neural networks (GNNs) encoder [Yang *et al.*, 2022; Yang *et al.*, 2023] as well as its perturbed version to generate two augmented views for contrast. AF-GCL [Wang *et al.*, 2022a] optimizes high-dimensional embedding distances using aggregated node features for positive and negative pair construction. SimGCL [Yu *et al.*, 2022] creates contrastive views by adding noises to the embeddings obtained by GNNs. Both the aforementioned methods and our method preserve the original graph structure information for contrastive training. The key difference is that our method does not require designing perturbed versions of encoders or embeddings, where manually tuning the perturbation ratio significantly affects model performance.

3 Preliminaries

Pre-training using GCL. Conventional GCL models use predefined GDA with frozen GDA parameters [Thakoor *et al.*, 2021; Zhu *et al.*, 2021]. As shown in Figure 1, it typically includes three main procedures: graph data augmentation, network encoding, and contrastive loss based optimization. Assume the input graph set $\mathcal{D}=\{\mathcal{G}_i|i=1,\cdots,n\}$ where $\mathcal{G}_i$

Notation	Description
$\mathcal{G}_i$	The i-th graph in input graph set $\mathcal{D}$.
$\tilde{\mathcal{G}}_i^{(t)}$	Augmented view of $\mathcal{G}_i$ at time t.
θ_A, θ_C	Parameter sets of Actor and Critic.
$M_i \in \mathbb{R}_+$	Number of node attributes in $\mathcal{G}_i$.
$N_i \in \mathbb{R}_+$	Number of nodes in $\mathcal{G}_i$.
$D \in \mathbb{R}_+$	Hidden size of embeddings.
$T \in \mathbb{R}_+$	Duration time of Actor or Critic.
$\boldsymbol{Z}_i, \tilde{\boldsymbol{Z}}_i^{(t)} \in \mathbb{R}^D$	Graph embeddings of $\mathcal{G}_i$ and $\tilde{\mathcal{G}}_i^{(t)}$.
$[\boldsymbol{X}_{\mathcal{G}_i}]_v \in \mathbb{R}^{M_i}$	Attribute vector of node v in $\mathcal{G}_i$.
$\tilde{\boldsymbol{\omega}}_i^{(t)} \in \mathbb{R}^{N_i \times N_i}$	GDA parameter matrix of $\tilde{\mathcal{G}}_i^{(t)}$.
$\tilde{\boldsymbol{Q}}_i^{(t)}, \tilde{\boldsymbol{V}}_i^{(t)} \in \mathbb{R}^{N_i \times N_i}$	Q-value matrix and V-value matrix.
$\tilde{\boldsymbol{A}}_i^{(t)} \in \mathbb{R}^{N_i \times N_i}$	Advantage-value matrix.
$f_{\boldsymbol{\phi}}(\cdot), g_{\boldsymbol{\varphi}}(\cdot)$	Encoder network, projection network.

Table 1: Frequently used notations.

denotes the i-th graph and n denotes the graph count. The augmented view $\tilde{\mathcal{G}}_i$ is obtained from $\mathcal{G}_i$ by a specific GDA operation $A_{\tilde{\boldsymbol{\omega}}}$ where $\tilde{\boldsymbol{\omega}}$ represents the GDA parameter matrix. For example, $A_{\tilde{\boldsymbol{\omega}}}$ can be edge removing with the removing ratio of 0.2 for each node. Next, $\mathcal{G}_i$, $\tilde{\mathcal{G}}_i$ are passed into encoder network $f_{\boldsymbol{\phi}}(\cdot)$ as well as projection network $g_{\boldsymbol{\varphi}}(\cdot)$ and then are extracted into two D-dimensional graph embeddings $\boldsymbol{Z}_i, \tilde{\boldsymbol{Z}}_i \in \mathbb{R}^D$. Finally, the parameters $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$ are updated by optimizing a predefined contrastive loss like InfoNCE [Zhu *et al.*, 2020; You *et al.*, 2020] between $\boldsymbol{Z}_i$ and $\tilde{\boldsymbol{Z}}_i$.

Fine-tuning on downstream tasks. GCL is generally followed by a certain downstream task such as graph classification [Sun *et al.*, 2020; Suresh *et al.*, 2021; Yin *et al.*, 2022]. When the training of GCL is completed, the graph embedding $\boldsymbol{Z}_i$ is generated from graph $\mathcal{G}_i$ by the trained encoder network and projection network. Then $\boldsymbol{Z}_i$ serves as the input of the downstream task's classifier such as a linear logistic regression model or an SVM model. For frequently used notations, we list and describe them in Table 1.

4 Methods

4.1 GRA Framework

Reinforcement learning involves no supervisor, and only a reward signal is used for an agent to determine if it is doing well or not [Wang *et al.*, 2022b]. From this point, reinforcement learning is suitable for characterizing an evolving view learner in GDA. We illustrate the proposed GRA framework in Figure 2. Specifically, there are three key components in this framework.

(C1) Min-max optimization of contrastive loss and cumulative reward. We design a mechanism of double-layer model learning: The outer layer is to minimize the InfoNCE loss $\mathcal{L}_{\text{NCE}}(\cdot)$ between two views by only updating $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$; the inner layer is to maximize the expected cumulative reward $R(\cdot)$ by only updating $\tilde{\boldsymbol{\omega}}_i$ $(i = 1, \cdots, n)$. The whole min-max optimization is formalized as follows:

$$\begin{aligned} &\min_{\boldsymbol{\phi},\boldsymbol{\varphi}} \mathcal{L}_{\text{NCE}}(g_{\boldsymbol{\varphi}}(f_{\boldsymbol{\phi}}(\mathcal{G}_i)),\ \ g_{\boldsymbol{\varphi}}(f_{\boldsymbol{\phi}}(A_{\tilde{\boldsymbol{\omega}}_i^*}(\mathcal{G}_i)))) \\ &\qquad \text{s.t.}\ \ \tilde{\boldsymbol{\omega}}_i^* = \arg\max_{\tilde{\boldsymbol{\omega}}_i} R(\tilde{\boldsymbol{\omega}}_i). \end{aligned} \tag{1}$$

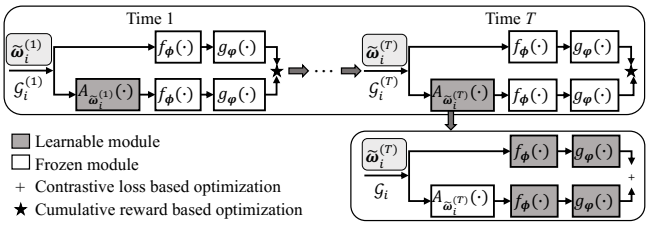

Figure 2: Forward propagation process of the GRA framework. For the backpropagation, it is operated twice in one epoch: One is at the end of the inner layer optimization and the other is at the end of the outer layer optimization.

(C2) Markov decision process for GDA. The GDA procedure in the GRA framework is continuous and learnable. The continuity ensures that the view learner adjusts the current GDA parameters according to the state at the previous time. Therefore, it makes the augmented view progressively evolve to be a learning-friendly view during model training. To be specific, the determination of GDA parameters is regarded as a Markov decision process. Given the present augmented view $\tilde{\mathcal{G}}_i^{(t)}$ at time t (note that one epoch includes T times), the next augmented view $\tilde{\mathcal{G}}_i^{(t+1)}$ at time $t+1$ is independent of any past view $\tilde{\mathcal{G}}_i^{(t')}$ where $t' < t$:

$$\mathbb{P}(\tilde{\mathcal{G}}_i^{(t+1)} | \tilde{\mathcal{G}}_i^{(t)}) = \mathbb{P}(\tilde{\mathcal{G}}_i^{(t+1)} | \tilde{\mathcal{G}}_i^{(1)}, \cdots, \tilde{\mathcal{G}}_i^{(t)}). \tag{2}$$

Then a Markov decision process for GDA is formulated as a triplet $(\mathcal{S}_{\text{state}}, \mathcal{S}_{\text{action}}, R)$: State set $\mathcal{S}_{\text{state}}$ is the set of all possible augmented views where each element represents a certain augmented view; action set $\mathcal{S}_{\text{action}}$ is the set of all possible augmentation operations where each element represents a certain augmentation operation; reward R is the total reward of the trajectory measuring the reward value generated by a series of states. For the state, action, and reward at time t, we denote them as $\tilde{\mathcal{G}}_i^{(t)}$, $A_{\tilde{\boldsymbol{\omega}}_i^{(t)}}(\cdot)$, and $R^{(t)}$, respectively. The augmentation operation $A_{\tilde{\boldsymbol{\omega}}_i^{(t)}}(\cdot)$ is uniquely determined by $\tilde{\boldsymbol{\omega}}_i^{(t)}$, and we elaborate on how to obtain $\tilde{\boldsymbol{\omega}}_i^{(t)}$ in Section 4.2. Moreover, to reduce the redundant information between two views as much as possible in GCL, we design $R^{(t)}$ as the negative mutual information (it can be calculated using the InfoNCE loss) between the embeddings $\boldsymbol{Z}_i^{(t)}, \tilde{\boldsymbol{Z}}_i^{(t)} \in \mathbb{R}^D$ at time t:

$$\begin{aligned} R^{(t)} &= -\mathcal{I}(\boldsymbol{Z}_i^{(t)}, \tilde{\boldsymbol{Z}}_i^{(t)}), \\ \boldsymbol{Z}_i^{(t)} = g_{\boldsymbol{\varphi}}(f_{\boldsymbol{\phi}}(\mathcal{G}_i^{(t)})),\ \ \tilde{\boldsymbol{Z}}_i^{(t)} &= g_{\boldsymbol{\varphi}}(f_{\boldsymbol{\phi}}(\tilde{\mathcal{G}}_i^{(t)})). \end{aligned} \tag{3}$$

(C3) Preserving graph structure information using edge reweighting. Most GCL methods employ edge removing or node deletion to achieve GDA, which disrupts the original graph structure information [You *et al.*, 2020; You *et al.*, 2022; Yin *et al.*, 2022; Suresh *et al.*, 2021]. Unlike previous strategies, we design *edge reweighting* for the GDA in GRA. Given the original edge weight matrix $\boldsymbol{E}_{\mathcal{G}_i} \in \mathbb{R}^{N_i \times N_i}$, the edge reweighting strategy Aug_{ERW} is formalized as follows:

$$\text{Aug}_{\text{ERW}} : \boldsymbol{E}_{\mathcal{G}_i} \mapsto \boldsymbol{E}_{\mathcal{G}_i} \circ \tilde{\boldsymbol{\omega}}_i^{(t)}, \tag{4}$$

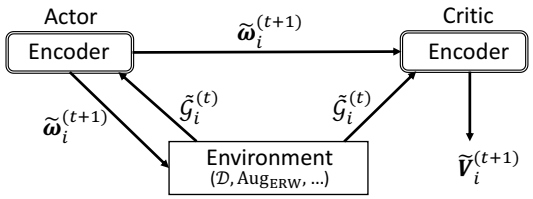

Figure 3: Illustration of the relationships among Actor, Critic, and environment. Here the environment includes the input graph set $\mathcal{D}$, edge reweighting strategy $\mathrm{Aug}_{\mathrm{ERW}}$, etc.

where $\tilde{\boldsymbol{\omega}}_i^{(t)} \in \mathbb{R}^{N_i \times N_i}$ denotes the GDA parameter matrix of $\tilde{\mathcal{G}}_i^{(t)}$ (the elements in this matrix correspond to edge weights) and $\circ$ represents the Hadamard product of two matrices. For an unweighted graph, $\boldsymbol{E}_{\mathcal{G}_i}$ is set to the adjacency matrix. Also, we initialize $\tilde{\boldsymbol{\omega}}_i^{(t)}$ as an all-ones matrix $\boldsymbol{J}_{N_i} \in \mathbb{R}^{N_i \times N_i}$ when $t = 0$, i.e., $\tilde{\boldsymbol{\omega}}_i^{(0)} = \boldsymbol{J}_{N_i}$.

For example, given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where node set $\mathcal{V} = \{v_1, v_2, v_3\}$ and edge set $\mathcal{E} = \{(v_1, v_2), (v_1, v_3)\}$, we define an edge index dictionary $E = \{e_1{:}(v_1, v_2), e_2{:}(v_1, v_3)\}$. Then $\tilde{\boldsymbol{\omega}}_i^{(t)}$ can be $[[1.0, 0.6, 0.8]^{\mathrm{T}}, [0.6, 1.0, 0.0]^{\mathrm{T}}, [0.8, 0.0, 1.0]^{\mathrm{T}}]$ where the numbers 0.6 and 0.8 represent the edge weights of e_1 and e_2 at time t, respectively. Edge weights act as coefficients for the message passing in GNNs, so the message along e_2 is more important than that along e_1 at time t.

4.2 GA2C Model

We present the details of GA2C, an instantiation of the GRA framework, including the model architecture, update of Critic, and update of Actor.

(I) Model architecture. Although we can design a vanilla policy gradient method by using the reward defined in Eq. (3) and a policy function updated by a sampled reward return, the sampled gradients likely indicate a wrong learning direction and thus lead to unstable training. To address this issue, we design the GA2C model containing two submodels of Actor and Critic. The relationships among the environment, Actor, and Critic are illustrated in Figure 3. Specifically, the Actor submodel contains the parameter set $\theta_A = \{\boldsymbol{W}_1, \cdots, \boldsymbol{W}_K\}$ where K denotes the GNN layer count, $\boldsymbol{W}_1 \in \mathbb{R}^{M_i \times N_i^2}$, and $\boldsymbol{W}_k \in \mathbb{R}^{N_i^2 \times N_i^2}$ $(k{=}2, \cdots, K)$. Given the node v's neighbor set $\mathcal{N}_i(v)$ as well as the edge weight $[\tilde{\boldsymbol{\omega}}_i^{(t)}]_{v,u}$ between nodes v and u, Actor receives the edge weight information in $\tilde{\mathcal{G}}_i^{(t)}$, and estimates $\tilde{\boldsymbol{\omega}}_i^{(t+1)}$ by K-layer GNNs:

$$\begin{aligned}
\tilde{\boldsymbol{\omega}}_i^{(t+1)} &= \mathrm{Sigmoid}(\mathrm{Reshape}(\boldsymbol{H}_i^{(t)})), \\
\boldsymbol{H}_i^{(t)} &= \mathrm{BN}(\mathrm{CC}(\mathrm{RD}(\left\{\boldsymbol{h}_{v,k}^{(t)} | v \in \tilde{\mathcal{G}}_i^{(t)}\right\}) | k{=}1, \cdots, K)), \\
\boldsymbol{h}_{v,k}^{(t)} &= \mathrm{MLP}_{\theta_A}^k\Big(\sum_{u \in \mathcal{N}_i(v) \cup \{v\}} \frac{[\tilde{\boldsymbol{\omega}}_i^{(t)}]_{v,u} \cdot \boldsymbol{h}_{u,k-1}^{(t)}}{d(v) \cdot d(u)}\Big),
\end{aligned} \tag{5}$$

where $\mathrm{Reshape}(\cdot)$ represents a function reshaping a vector $\boldsymbol{H}_i^{(t)} \in \mathbb{R}^{N_i^2}$ into a matrix $\tilde{\boldsymbol{\omega}}_i^{(t+1)} \in \mathbb{R}^{N_i \times N_i}$, $\mathrm{BN}(\cdot)$ the batch normalization layer, $\mathrm{CC}(\cdot)$ the concatenation layer, $\mathrm{RD}(\cdot)$ the readout layer calculating the temporary variable of graph $\tilde{\mathcal{G}}_i$, $d(\cdot)$ the node degree, and $\boldsymbol{h}_{v,k}^{(0)} = [\boldsymbol{X}_{\mathcal{G}_i}]_v$ the attribute vector of node v. Since the GDA parameter matrix $\tilde{\boldsymbol{\omega}}_i^{(t+1)}$ is learned automatically based on the previous state $\tilde{\mathcal{G}}_i^{(t)}$, it ensures that the augmented view evolves progressively in the training.

As an evaluator of the actions taken by Actor, the Critic submodel contains the parameter set $\theta_C = \{\boldsymbol{W}_1', \cdots, \boldsymbol{W}_K'\}$ where $\boldsymbol{W}_1' \in \mathbb{R}^{M_i \times N_i^2}$, and $\boldsymbol{W}_k' \in \mathbb{R}^{N_i^2 \times N_i^2}$ $(k{=}2, \cdots, K)$. Critic receives $\tilde{\mathcal{G}}_i^{(t)}$ and $\tilde{\boldsymbol{\omega}}_i^{(t+1)}$, and then estimates the V-value matrix $\tilde{\boldsymbol{V}}_i^{(t+1)}$ by K-layer GNNs. Specifically, a new view $\tilde{\mathcal{G}}_i^{(t+1)}$ at time $t{+}1$ is generated by operating edge reweighting on $\mathcal{G}_i^{(t)}$:

$$\tilde{\mathcal{G}}_i^{(t+1)} = A_{\tilde{\boldsymbol{\omega}}_i^{(t+1)}}(\mathcal{G}_i^{(t)}), \tag{6}$$

where the input graph at the initial time $\tilde{\mathcal{G}}_i^{(1)} = \mathcal{G}_i$. Then the Critic submodel estimates $\tilde{\boldsymbol{V}}_i^{(t+1)}$ based on the edge weight information in $\tilde{\mathcal{G}}_i^{(t+1)}$. Without loss of generality, $\tilde{\boldsymbol{V}}_i^{(t)}$ is estimated as follows:

$$\begin{aligned}
\tilde{\boldsymbol{V}}_i^{(t)} &= \mathrm{Reshape}(\boldsymbol{M}_i^{(t)}), \\
\boldsymbol{M}_i^{(t)} &= \mathrm{BN}(\mathrm{CC}(\mathrm{RD}(\left\{\boldsymbol{m}_{v,k}^{(t)} | v \in \tilde{\mathcal{G}}_i^{(t)}\right\}) | k{=}1, \cdots, K)), \\
\boldsymbol{m}_{v,k}^{(t)} &= \mathrm{MLP}_{\theta_C}^k\Big(\sum_{u \in \mathcal{N}_i(v) \cup \{v\}} \frac{[\tilde{\boldsymbol{\omega}}_i^{(t)}]_{v,u} \cdot \boldsymbol{m}_{u,k-1}^{(t)}}{d(v) \cdot d(u)}\Big),
\end{aligned} \tag{7}$$

where $\boldsymbol{m}_{v,k}^{(t)}$ is initialized as the attribute vector of node v, i.e., $\boldsymbol{m}_{v,k}^{(1)} = [\boldsymbol{X}_{\mathcal{G}_i}]_v$. We train Actor and Critic using the gradient descent algorithm. The following is a detailed description about the updates of Actor and Critic.

(II) Update of Critic. The optimization objective of Critic is to maximize $\mathcal{J}_C = \sum_{t=1}^{T} (\tilde{\boldsymbol{A}}_i^{(t)})^2$ and the gradient w.r.t. θ_C is computed as follows:

$$\nabla_{\theta_C} \mathcal{J}_C = \frac{d}{d\theta_C} \sum_{t=1}^{T} (\tilde{\boldsymbol{A}}_i^{(t)})^2, \tag{8}$$

where the advantage-value matrix $\tilde{\boldsymbol{A}}_i^{(t)}$ denotes the difference between the actual outcome (i.e., Q-value matrix $\tilde{\boldsymbol{Q}}_i^{(t)}$) brought by the GDA operation $A_{\tilde{\boldsymbol{\omega}}_i^{(t)}}(\cdot)$ and the expected outcome (i.e., V-value matrix $\tilde{\boldsymbol{V}}_i^{(t)}$) at the given state $\tilde{\mathcal{G}}_i^{(t)}$. Because calculating $\tilde{\boldsymbol{Q}}_i^{(t)}$ requires a new group of GNNs, we estimate $\tilde{\boldsymbol{A}}_i^{(t)}$ with temporal difference error [Bengio *et al.*, 2020] to simplify computations:

$$\tilde{\boldsymbol{A}}_i^{(t)} = \tilde{\boldsymbol{Q}}_i^{(t)} - \tilde{\boldsymbol{V}}_i^{(t)} = R^{(t+1)} + \gamma \tilde{\boldsymbol{V}}_i^{(t+1)} - \tilde{\boldsymbol{V}}_i^{(t)}, \tag{9}$$

where $R^{(t+1)}$ is defined in Eq. (3).

(III) Update of Actor. The optimization objective of Actor is to maximize $\mathcal{J}_A = \sum_{t=1}^{T} \log \pi_i(\tilde{\boldsymbol{\omega}}_i^{(t+1)} | \tilde{\mathcal{G}}_i^{(t)}) \cdot \tilde{\boldsymbol{A}}_i^{(t)}$ and the gradient w.r.t. θ_A is computed as follows:

Algorithm 1 Training process of GA2C

Input: Graph set $\mathcal{D}$, epoch count N_e, duration time T, Actor's learning rate α_A, Critic's learning rate α_C, encoder network's learning rate α_E, and learnable parameters $\boldsymbol{\phi}, \boldsymbol{\varphi}, \theta_A, \theta_C$.

Output: Trained encoder network $f_{\boldsymbol{\phi}}$ and projection network $g_{\boldsymbol{\varphi}}$.

Initialize $\boldsymbol{\phi}$, $\boldsymbol{\varphi}$, θ_A, and θ_C with the Glorot uniform initializer;
for $n \in [1, N_e]$ **do**
  for $\mathcal{G}_i \in \mathcal{D}$ **do**
    Initialization: $\tilde{\mathcal{G}}_i^{(1)} \leftarrow \mathcal{G}_i$, $\tilde{\boldsymbol{\omega}}_i^{(0)} \leftarrow \boldsymbol{J}_{N_i}$;
    for $t \in [1, T]$ **do**
      Calculate $\tilde{\boldsymbol{\omega}}_i^{(t)}$ based on Eq. (5) and take action based on Eq. (6);
      Calculate $\tilde{\boldsymbol{V}}_i^{(t)}$ based on Eq. (7);
      Calculate $\tilde{\boldsymbol{A}}_i^{(t)}$ based on Eq. (9);
    end for
    Update θ_A: $\theta_A \leftarrow \theta_A - \alpha_A \nabla_{\theta_A} \mathcal{J}_A$ based on Eq. (10);
    Update θ_C: $\theta_C \leftarrow \theta_C - \alpha_C \nabla_{\theta_C} \mathcal{J}_C$ based on Eq. (8);
    Calculate $\mathcal{L}_{\text{NCE}}$ in Eq. (1) with $\tilde{\boldsymbol{\omega}}_i^* = \tilde{\boldsymbol{\omega}}_i^{(T)}$;
    Update the parameter $\boldsymbol{\phi}$ in $f_{\boldsymbol{\phi}}$: $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha_E \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\text{NCE}}$;
    Update the parameter $\boldsymbol{\varphi}$ in $g_{\boldsymbol{\varphi}}$: $\boldsymbol{\varphi} \leftarrow \boldsymbol{\varphi} - \alpha_E \nabla_{\boldsymbol{\varphi}} \mathcal{L}_{\text{NCE}}$;
  end for
end for
return $f_{\boldsymbol{\phi}}$ and $g_{\boldsymbol{\varphi}}$.

$$\nabla_{\theta_A} \mathcal{J}_A = \sum_{t=1}^{T} \frac{d \log \pi_i(\tilde{\boldsymbol{\omega}}_i^{(t+1)} | \tilde{\mathcal{G}}_i^{(t)})}{d \theta_A} \cdot \tilde{\boldsymbol{A}}_i^{(t)}, \tag{10}$$

where the policy $\pi_i(\tilde{\boldsymbol{\omega}}_i^{(t+1)} | \tilde{\mathcal{G}}_i^{(t)})$ represents the probability of selecting $A_{\tilde{\boldsymbol{\omega}}_i^{(t+1)}}(\cdot)$ as the action given the state $\tilde{\mathcal{G}}_i^{(t)}$. We adopt a stochastic function to generate all policy values. The pseudocode for the complete training process of GA2C is shown in Algorithm 1.

4.3 Time Complexity Analysis

Training. Let L be the layer count in the encoder network, N_{avg} the average node count in each augmented graph, and M_{avg} the average attribute count in each augmented graph. We conduct batch learning where each batch has B graphs and employ a *triggering learning mechanism* for GDA, where GDA parameters are updated once every T epochs. Therefore, the total time complexity of GA2C training in each batch per epoch is $O(N_e B(TK+L) N_{\text{avg}} M_{\text{avg}} D/T) = O(N_e B(K+L/T) N_{\text{avg}} M_{\text{avg}} D)$ where N_e, T, and D denote the epoch number, duration time, and hidden size, respectively.

Inference. In model inference, the raw graphs are directly used as the input data of the encoder and projection networks. Therefore, the reinforced GDA procedure does not add any extra complexity to model inference, and the inference complexity of GA2C is identical to the state-of-the-art GCL models such as AD-GCL [Suresh *et al.*, 2021] and SimGRACE [Xia *et al.*, 2022]. Then given a test graph whose node count and attribute count are respectively denoted as N_{test} and M_{test}, the time complexity of GA2C inference is $O(LN_{\text{test}}M_{\text{test}}D)$.

Dataset	Type	#Graph	#AN	#AE	#Class
NCI1	Biochemistry	4,110	29.87	32.30	2
PROTEINS	Biochemistry	1,113	39.06	72.82	2
MUTAG	Biochemistry	188	17.93	19.79	2
DD	Biochemistry	1,178	284.32	715.66	2
REDDIT-B	Social graphs	2,000	429.6	497.75	2
REDDIT-M-5K	Social graphs	4,999	508.8	594.87	5
IMDB-B	Social graphs	1,000	19.8	96.53	2
COLLAB	Social graphs	5,000	74.49	2,457.78	3
GITHUB	Social graphs	12,725	113.79	234.64	2
Dataset	**Type**	**#Graph**	**#AN**	**#AE**	**#Task**
ToxCast	Physiology	8,576	18.8	19.3	617
BBBP	Physiology	2,039	24.1	26.0	1
MUV	Biophysics	93,087	24.2	26.3	17
BACE	Biophysics	1,513	34.1	36.9	1

Table 2: The statistics of the experimental datasets. "#AN" and "#AE" represent the average node count and average edge count in each graph, respectively.

5 Experiments

5.1 Configurations

Datasets. The experimental datasets are from TU datasets [Morris *et al.*, 2020] and Open Graph Benchmark (OGB) datasets [Hu *et al.*, 2020a]. They cover four types of graphs including biochemical molecules (NCI1, PROTEINS, MUTAG, and DD), social graphs (REDDIT-B, REDDIT-M-5K, IMDB-B, COLLAB, and GITHUB), physiology (ToxCast and BBBP), and biophysics (MUV and BACE). The statistics of these datasets are shown in Table 2. As with previous work, we report the accuracy results on TU datasets and the ROC-AUC results on OGB datasets. In addition, we use ZINC-2M [Hu *et al.*, 2020b] as the pre-training dataset in transfer learning. In ZINC-2M, there are a total of 2 million unlabeled molecules, with each molecule graph having an average node count of 26.62 and an average degree of 57.72.

Baselines. We compare GA2C with different types of baselines: (i) In unsupervised learning, the baselines include kernel-based methods such as graphlet kernel (GL) [Shervashidze *et al.*, 2009], Weisfeiler-Lehman sub-tree kernel (WL) [Shervashidze *et al.*, 2011], and deep graph kernel (DGK) [Yanardag and Vishwanathan, 2015]. In addition, we compare unsupervised graph representation learning methods, which include node2vec [Grover and Leskovec, 2016], sub2vec [Adhikari *et al.*, 2018], and graph2vec [Narayanan *et al.*, 2017]. Furthermore, we compare state-of-the-art GCL methods, namely InfoGraph [Sun *et al.*, 2020], GraphCL [You *et al.*, 2020], JOAOv2 [You *et al.*, 2021], AD-GCL [Suresh *et al.*, 2021], AutoGCL [Yin *et al.*, 2022], and SimiGRACE [Xia *et al.*, 2022]. SimiGRACE is an augmentation-free method, while JOAOv2, AD-GCL, and AutoGCL adopt learnable GDA. (ii) In transfer learning, we also compare Infomax [Velickovic *et al.*, 2019], EdgePred [Hu *et al.*, 2020b], AttrMasking [Hu *et al.*, 2020b], and ContextPred [Hu *et al.*, 2020b]. (iii) In semi-supervised learning, we also compare GCA [Zhu *et al.*, 2021] that uses precomputed methods for GDA.

Method	MUTAG	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M-5K	AR ↓
GL	81.66±2.11	-	-	-	-	65.87±0.98	77.34±0.18	41.01±0.17	10.0
WL	80.72±3.00	72.92±0.56	-	80.01±0.50	-	72.30±3.44	68.82±0.41	46.06±0.21	7.5
DGK	87.44±2.72	73.30±0.82	-	80.31±0.46	-	66.96±0.56	78.04±0.39	41.27±0.18	7.2
node2vec	72.63±10.20	57.49±3.57	-	54.89±1.61	-	-	-	-	11.0
sub2vec	61.05±15.80	53.03±5.55	-	52.84±1.47	-	55.26±1.54	71.48±0.41	36.68±0.42	11.8
graph2vec	83.15±9.25	73.30±2.05	-	73.22±1.81	-	71.10±0.54	75.78±1.03	47.86±0.26	8.5
InfoGraph	89.01±1.13	74.44±0.31	72.85±1.78	76.20±1.06	70.65±1.13	73.03±0.87	82.50±1.42	53.46±1.03	5.1
GraphCL	86.80±1.34	74.39±0.45	**78.62±0.40**	77.87±0.41	71.36±1.15	71.14±0.44	89.53±0.84	**55.99±0.28**	4.5
AD-GCL	88.62±1.27	75.04±0.48	75.38±0.41	75.77±0.50	71.95±0.79	71.49±0.98	84.16±1.18	55.47±0.72	5.0
JOAOv2	-	71.25±0.85	66.91±1.75	72.99±0.75	70.40±2.21	71.60±0.86	78.35±1.38	45.57±2.86	7.7
AutoGCL	89.28±0.89	75.76±0.38	77.08±0.53	79.79±0.42	71.85±1.02	72.24±0.42	88.46±0.71	55.83±0.53	3.4
SimGRACE	89.01±1.31	75.35±0.09	77.44±1.11	79.12±0.44	71.72±0.82	71.30±0.77	89.51±0.89	55.91±0.34	3.9
GA2C	**90.34±0.39**	**76.03±0.16**	78.10±0.69	**80.62±0.39**	**72.13±0.34**	**73.88±0.39**	**89.62±0.95**	55.93±0.64	**1.3**

Table 3: Unsupervised learning results (Accuracy, %) on TU datasets. We repeat the experiment of GA2C five times using different random seeds, and then report the mean as well as standard deviation as final results. The best result is **bolded** and the runner-up is underlined. AR denotes average rank and - indicates that results are not available in published papers.

Hyperparameter settings. (i) In unsupervised learning, for REDDIT-B and REDDIT-M-5K, we use a 5-layer graph isomorphism network (GIN) [Xu *et al.*, 2019] with a hidden size of 128 as our encoder network and a 2-layer MLP with a hidden size of 128 as our projection network (training 150 epochs in total); for the other datasets, we use a 3-layer GIN with a hidden size of 32 as our encoder network and a 2-layer MLP with a hidden size of 128 as our projection network (training 60 epochs in total). The submodel of Actor or Critic is implemented as the same network architecture of the encoder network. The learning rates for the Actor, Critic, and encoder network are consistently set to 1e-3. For downstream tasks, we adopt an SVM as the classifier and perform 10-fold cross-validation on each dataset. For each fold, we allocate 90% of the total data as the unlabeled data for contrastive pre-training, and the remaining 10% as the labeled testing data. (ii) In transfer learning, we pre-train GA2C on the ZINC-2M dataset [Hu *et al.*, 2020b] with 10 epochs and set the hidden sizes of the encoder network (3-layer GIN) and projection network (2-layer MLP) as 300 and 64, respectively. Then we fine-tune GA2C on a specific OGB dataset with 100 epochs. The split ratio of the training, validation, and testing sets is 8:1:1. (iii) In semi-supervised learning, we use a 3-layer GIN with a hidden size of 128 as the submodel of Actor or Critic and use a 3-layer ResGCN [Chen *et al.*, 2019] with a hidden size of 128 as the classifier. We perform 10-fold cross-validation on each dataset. For each fold, we allocate 80% of the total data as unlabeled data, 10% as labeled training data, and the remaining 10% as labeled testing data.

5.2 Results and Analysis

Results of unsupervised learning. We report the performance results of GA2C and published results of baselines on TU datasets in Table 3. We can see that GCL methods outperform kernel-based methods and unsupervised graph representation learning methods. Furthermore, view-learnable GCL methods like AutoGCL further enhance the performance of GCL methods, performing the best in all baselines. As an augmentation-free method, SimGRACE preserves the original graph structure information and also performs well. Compared to these baselines, GA2C performs better, with the best average ranking of 1.3. GA2C achieves the first place on 6 out of 8 datasets and the second place on 2 of them. It is mainly attributed to the design of graph reinforcement augmentation, which not only preserves the structure information of the original graph but also maintains the characteristics of progressive evolution in augmented views.

Method	BBBP	ToxCast	MUV	BACE	AR ↓
No Pre-train	65.8±4.5	63.4±0.6	71.8±2.5	70.1±5.4	9.0
Infomax	68.8±0.8	62.7±0.4	75.3±2.5	75.9±1.6	7.0
EdgePred	67.3±2.4	64.1±0.6	74.1±2.1	79.9±0.9	5.3
AttrMasking	64.3±2.8	**64.2±0.5**	74.7±1.4	79.3±1.6	5.5
ContextPred	68.0±2.0	63.9±0.6	75.8±1.7	79.6±1.2	4.5
GraphCL	69.6±0.6	62.4±0.5	69.8±2.6	75.3±1.4	9.0
JOAOv2	71.3±0.9	63.1±0.4	73.6±1.0	75.4±1.2	6.5
AD-GCL	70.0±1.0	63.0±0.7	72.3±1.6	78.5±0.8	7.0
AutoGCL	73.3±0.7	63.4±0.3	75.8±1.3	83.2±1.1	2.8
SimGRACE	71.2±0.8	63.3±0.5	69.4±1.2	74.6±0.7	8.0
GA2C	**74.0±1.0**	**64.2±0.2**	**79.7±0.4**	**83.9±0.4**	**1.0**

Table 4: Transfer learning results (ROC-AUC, %) on OGB datasets.

Results of transfer learning. We use a method called "No Pre-train" as a control method, which directly trains on each dataset without pre-training. Based on the results in Table 4, we can see that most of the pre-training methods outperform the ones without pre-training. Among all the baselines, AutoGCL demonstrates the best overall performance, indicating the superiority of learnable graph data augmentation over the frozen one. In contrast, GA2C performs better than AutoGCL and achieves the best performance on all datasets. For example, GA2C shows a significant improvement in MUV, with an increase of 3.9%. The comparison in transfer learning validates the effectiveness of the GRA framework and its instantiation model GA2C.

Results of semi-supervised learning. We calculate the average ranking for each method apart from "Full Data" and report the total results in Table 5. "Full Data" adopts 90% data for training and 10% for testing; "10% Data" adopts 10%

Method	PROTEINS	DD	NCI1	COLLAB	IMDB-B	REDDIT-B	REDDIT-M-5K	GITHUB	AR ↓
Full Data	78.25±1.61	80.73±3.78	83.65±1.16	83.44±0.77	76.60±4.20	89.95±2.06	55.59±2.24	66.89±1.04	-
10% Data	69.72±6.71	74.36±5.86	75.16±2.07	74.34±2.00	64.80±4.92	76.75±5.60	49.71±3.20	61.05±1.57	6.6
10% GCA	73.85±5.56	76.74±4.09	68.73±2.36	74.32±2.30	**73.70±4.88**	77.15±6.96	32.95±10.89	59.24±3.21	6.0
10% GraphCL	74.21±4.50	76.65±5.12	73.16±2.90	75.50±2.15	68.10±5.15	78.05±2.65	48.09±1.74	63.51±1.02	5.4
10% JOAOv2	73.31±0.48	75.81±0.73	74.86±0.39	75.53±0.18	68.30±4.08	88.79±0.65	52.71±0.28	66.66±0.60	4.6
10% AD-GCL	73.96±0.47	77.91±0.73	75.18±0.31	75.82±0.26	72.48±1.15	90.10±0.15	53.49±0.28	-	2.7
10% AutoGCL	75.65±2.40	77.50±4.41	73.75±2.25	**77.16±1.48**	71.90±4.79	79.80±3.47	49.91±2.70	62.46±1.51	3.8
10% SimGRACE	74.03±0.51	76.48±0.52	74.60±0.41	74.74±0.28	71.30±0.77	88.86±0.62	**53.97±0.64**	-	4.3
10% GA2C	**76.10±2.23**	**79.62±4.65**	**75.84±1.50**	76.16±2.00	**73.70±4.27**	**90.35±2.49**	52.39±1.88	**68.90±1.62**	**1.5**

Table 5: Semi-supervised learning results on TU datasets. We mark the best result and runner-up, excluding Full Data.

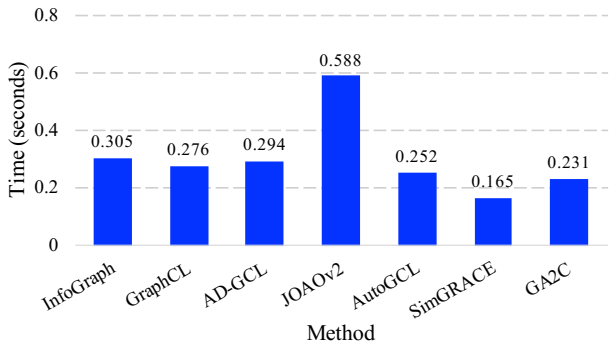

Figure 4: Comparison of training time per epoch.

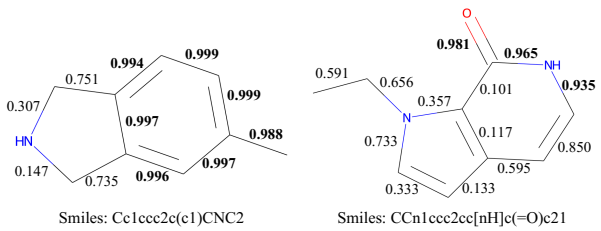

Figure 5: Two example molecular graphs in ZINC-2M. The bolded numbers represent the important chemical bonds learned by GA2C.

data for training and does not include the pre-training step; the remaining methods adopt 10% data for training and include the pre-training step. In this comparison, GA2C, AD-GCL, and AutoGCL achieve the first, second, and third positions, respectively. The above three methods use learnable GDA, which verifies its advantage over frozen GDA. AD-GCL performs the best among baselines, with an average ranking of 2.7. It achieves the second position on 5 out of 8 datasets. Our proposed model GA2C achieves the first position on 6 out of 8 datasets, with an average ranking of 1.5. For example, on DD and GITHUB, GA2C outperforms the runner-ups (AD-GCL and JOAOv2, respectively) by 1.71% and 2.24%, respectively. The above analysis demonstrates that GA2C surpasses state-of-the-art GCL methods in either unsupervised, transfer, or semi-supervised learning, confirming the effectiveness of graph reinforcement augmentation.

Comparison of training time. Using the same batch size of 32, we compare the average training time per epoch among different GCL methods and GA2C. The results are shown in Figure 4. Overall, in terms of training efficiency, GA2C is in the same order of magnitude as these methods. The least time-consuming method is SimGRACE because it uses encoder perturbation instead of GDA to reduce training time. The second efficient method is GA2C which adopts a triggering learning mechanism for GDA. Such a mechanism in GA2C makes its training efficiency second only to SimGRACE and more efficient than other GCL methods. We observe that JOAOv2 has the longest training time per epoch among these methods, indicating it spends excessive time on data augmentation selection during bi-level optimization.

Case studies. We conduct case studies to qualitatively demonstrate the effectiveness of the learnable GDA in our proposed framework. In the pre-training of transfer learning, we select two example chemical molecules from ZINC-2M [Hu *et al.*, 2020b]. Figure 5 shows the edge weights of chemical bonds in the augmented views learned by GA2C at the final training epoch. For the edge weights greater than 0.9, we highlight them in bold. For the left molecule, we find that the edge weights of the six carbon-carbon single bonds on the benzene ring are all relatively high ($>$0.99). It indicates that the benzene ring in this molecule significantly influences its chemical properties. For the right molecule, we observe that the edge weights of the carbon-oxygen double bond and certain carbon-nitrogen single bonds are high ($>$0.93). The carbon-oxygen double bond is an important chemical bond in ketones and significantly influences the chemical properties of the molecule. Furthermore, two carbon-nitrogen single bonds in this molecule serve to connect the secondary amine group and ring structure, so they also have high edge weights.

6 Conclusion

In this paper, we propose a novel graph reinforcement augmentation framework, ensuring that the augmented view evolves progressively and preserves the original graph structure information to promote the performance of GCL. Based on this framework, we design a new GCL model called GA2C to learn graph embeddings in an unsupervised manner. Through extensive experiments, we verify that GA2C outperforms the state-of-the-art GCL models on either unsupervised, semi-supervised, or transfer learnings. By case studies, we demonstrate that GA2C generates meaningful augmented views. In the future, we plan to apply GA2C to heterogeneous graphs.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. 62372264). Chaokun Wang is the corresponding author.

References

[Adhikari *et al.*, 2018] Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B. Aditya Prakash. Sub2vec: Feature learning for subgraphs. In *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference*, pages 170–182, Melbourne, VIC, Australia, June 2018. Springer.

[Ang *et al.*, 2021] Andersen Man Shun Ang, Jianzhu Ma, Nianjun Liu, Kun Huang, and Yijie Wang. Fast projection onto the capped simplex with applications to sparse regression in bioinformatics. In *Thirty-fifth Annual Conference on Neural Information Processing Systems*, pages 9990–9999, Virtual Event, December 2021. MIT Press.

[Bengio *et al.*, 2020] Emmanuel Bengio, Joelle Pineau, and Doina Precup. Interference and generalization in temporal difference learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 767–777, Virtual Event, July 2020. PMLR.

[Chen *et al.*, 2019] Ting Chen, Song Bian, and Yizhou Sun. Are powerful graph neural nets necessary? a dissection on graph classification. *CoRR*, abs/1905.04579, 2019.

[Cui *et al.*, 2019] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Trans. Knowl. Data Eng.*, 31(5):833–852, 2019.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, San Francisco, CA, USA, August 2016. ACM.

[Hu *et al.*, 2020a] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[Hu *et al.*, 2020b] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, April 2020. OpenReview.net.

[Ju *et al.*, 2023] Mingxuan Ju, Yujie Fan, Chuxu Zhang, and Yanfang Ye. Let graph be the go board: gradient-free node injection attack for graph neural networks via reinforcement learning. In *Thirty-Seventh AAAI Conference on Artificial Intelligence*, pages 4383–4390, Washington, DC, USA, February 2023. AAAI Press.

[Liu *et al.*, 2022] Ziyang Liu, Chaokun Wang, Hao Feng, Lingfei Wu, and Liqun Yang. Knowledge distillation based contextual relevance matching for e-commerce product search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing - Industry Track*, Abu Dhabi, UAE, December 2022. Association for Computational Linguistics.

[Liu *et al.*, 2023] Ziyang Liu, Chaokun Wang, Yunkai Lou, and Hao Feng. Fast unsupervised graph embedding via graph zoom learning. In *39th IEEE International Conference on Data Engineering*, pages 2551–2564, Anaheim, CA, USA, April 2023. IEEE.

[Morris *et al.*, 2020] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020.

[Munikoti *et al.*, 2023] Sai Munikoti, Deepesh Agarwal, Laya Das, Mahantesh Halappanavar, and Balasubramaniam Natarajan. Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2023.

[Narayanan *et al.*, 2017] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. In *Proceedings of the 13th International Workshop on Mining and Learning with Graphs (MLG)*, 2017.

[Shervashidze *et al.*, 2009] Nino Shervashidze, S. V. N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten M. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, pages 488–495, Clearwater Beach, Florida, USA, April 2009. JMLR.org.

[Shervashidze *et al.*, 2011] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12:2539–2561, 2011.

[Sun *et al.*, 2020] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, April 2020. OpenReview.net.

[Suresh *et al.*, 2021] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. In *Thirty-seventh Annual Conference on Neural Information Processing Systems*, Virtual Event, December 2021. MIT Press.

[Thakoor *et al.*, 2021] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Velickovic, and Michal Valko. Bootstrapped representation learning on graphs. *ICLR Workshop on Geometrical and Topological Representation Learning*, 2021.

[Tschannen *et al.*, 2020] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On

mutual information maximization for representation learning. In *8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, April 2020. OpenReview.net.

[Velickovic *et al.*, 2019] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. In *7th International Conference on Learning Representations*, New Orleans, LA, USA, May 2019. OpenReview.net.

[Wang *et al.*, 2018] Zhen Wang, Marko Jusup, Lei Shi, Joung-Hun Lee, Yoh Iwasa, and Stefano Boccaletti. Exploiting a cognitive bias promotes cooperation in social dilemma experiments. *Nature communications*, 9(1):2954, 2018.

[Wang *et al.*, 2022a] Haonan Wang, Jieyu Zhang, Qi Zhu, and Wei Huang. Augmentation-free graph contrastive learning. *arXiv preprint arXiv:2204.04874*, 2022.

[Wang *et al.*, 2022b] Zhen Wang, Chunjiang Mu, Shuyue Hu, Chen Chu, and Xuelong Li. Modelling the dynamics of regret minimization in large agent populations: a master equation approach. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 534–540, Vienna, Austria, July 2022. Morgan Kaufmann.

[Wu *et al.*, 2023] Cheng Wu, Chaokun Wang, Jingcao Xu, Ziyang Liu, Kai Zheng, Xiaowei Wang, Yang Song, and Kun Gai. Graph contrastive learning with generative adversarial network. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2721–2730, Long Beach, CA, USA, August 2023. ACM.

[Xia *et al.*, 2022] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z. Li. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the Web Conference 2022*, pages 1070–1079, Lyon, France, April 2022. ACM.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations*, New Orleans, LA, USA, May 2019. OpenReview.net.

[Xu *et al.*, 2023] Jingcao Xu, Chaokun Wang, Cheng Wu, Yang Song, Kai Zheng, Xiaowei Wang, Changping Wang, Guorui Zhou, and Kun Gai. Multi-behavior self-supervised learning for recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 496–505, Taipei, Taiwan, July 2023. ACM.

[Yanardag and Vishwanathan, 2015] Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374, Sydney, NSW, Australia, August 2015. Springer.

[Yang *et al.*, 2022] Liang Yang, Wenmiao Zhou, Weihang Peng, Bingxin Niu, Junhua Gu, Chuan Wang, Xiaochun Cao, and Dongxiao He. Graph neural networks beyond compromise between attribute and topology. In *Proceedings of the Web Conference 2022*, pages 1127–1135, Lyon, France, April 2022. ACM.

[Yang *et al.*, 2023] Liang Yang, Qiuliang Zhang, Runjie Shi, Wenmiao Zhou, Bingxin Niu, Chuan Wang, Xiaochun Cao, Dongxiao He, Zhen Wang, and Yuanfang Guo. Graph neural networks without propagation. In *Proceedings of the Web Conference 2023*, pages 469–477, Austin, TX, USA, April 2023. ACM.

[Yin *et al.*, 2022] Yihang Yin, Qingzhong Wang, Siyu Huang, Haoyi Xiong, and Xiang Zhang. Autogcl: Automated graph contrastive learning via learnable view generators. In *Thirty-Sixth AAAI Conference on Artificial Intelligence*, pages 8892–8900, Virtual Event, February 2022. AAAI Press.

[You *et al.*, 2020] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Thirty-sixth Annual Conference on Neural Information Processing Systems*, Virtual Event, December 2020. MIT Press.

[You *et al.*, 2021] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *International Conference on Machine Learning*, pages 12121–12132, Virtual Event, July 2021. PMLR.

[You *et al.*, 2022] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. Bringing your own view: Graph contrastive learning without prefabricated data augmentations. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1300–1309, Tempe, AZ, USA, February 2022. ACM.

[Yu *et al.*, 2022] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1294–1303, Madrid, Spain, July 2022. ACM.

[Zhu *et al.*, 2020] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. In *ICML Workshop on Graph Representation Learning and Beyond*, 2020.

[Zhu *et al.*, 2021] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pages 2069–2080, Ljubljana, Slovenia, April 2021. ACM.