GRAPH CONTRASTIVE LEARNING WITH REINFORCED AUGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

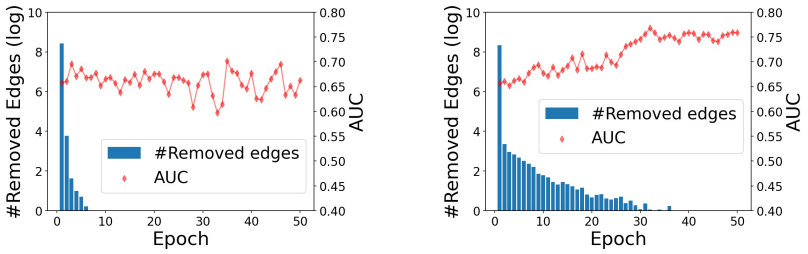
Graph contrastive learning (GCL), designing contrastive objectives to learn embeddings from augmented graphs, has become a prevailing method for learning embeddings from graphs in an unsupervised manner. As an important procedure in GCL, graph data augmentation (GDA) directly affects the model performance on the downstream task. Currently, there are three types of GDA styles: trial-and-error, precomputed method, and adversarial method. However, these strategies ignore the connection between the two consecutive augmentation results because GDA is regarded as an independent process. In this paper, we regard the GDA in GCL as a Markov decision process. Based on this point, we propose a reinforced method, i.e., the fourth type of GDA strategy, using a novel Graph Advantage Actor-Critic (GA2C) model for GCL. On 23 graph datasets, the experimental results verify that GA2C outperforms the SOTA GCL models on a series of downstream tasks such as graph classification, node classification, and link prediction.

1 INTRODUCTION

Graph representation learning aims to extract low-dimensional representation vectors from the graph data. 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 can be applied to a series of fields including bioinformatics Ang et al. (2021); Wang et al. (2021), social networks Matakos et al. (2022); Shen et al. (2022), and recommender systems Xu et al. (2022); Carroll et al. (2022). Due to the sparsity of labeling information in the real world, learning representations from graphs in a self-supervised manner has become a new research highlight Wu et al. (2021). Graph contrastive learning (GCL), as an important member of self-supervised representation learning on graphs, has achieved state-of-the-art (SOTA) performance on many downstream tasks Yu et al. (2022b); Suresh et al. (2021).

Graph data augmentation (GDA) is an important procedure in GCL. It defines the augmented views (i.e., the specific contents of input data for contrastive learning) and affects the model performance on the downstream task directly. Typically, GDA strategies include edge adding, edge removing, attribute masking, and so on. Prior work designs three types of GDA styles in total, including trial-and-error, precomputed method, and adversarial method. In the trial-and-error (e.g., GRACE Zhu et al. (2020) and BGRL Thakoor et al. (2021)) and precomputed method (e.g., GCA Zhu et al. (2021)), GDA is frozen in the training. While in the adversarial method (e.g., AD-GCL Suresh et al. (2021)), GDA is learnable and can be optimized by the view learner in the training.

Motivation: The Training Bottleneck in the Learnable GDA. Currently, GCL with learnable GDA unleashes the potential of contrastive learning and has achieved SOTA performance in the downstream task. Then a new potential question arises: *how does a good augmented view evolve to promote graph contrastive learning?* As an example, we plot the results of AD-GCL in Figure 1(a) where the x-axis, left y-axis, and right y-axis represents the epoch index, logarithmic number of removed edges, and area under the ROC curve (AUC) on the test data, respectively. We can see that the evolution of GDA does not promote graph contrastive learning well: on the one hand, after the sixth epoch, GDA does not work anymore; on the other hand, the AUC performance on the test data is unstable and has high variance. It indicates that the current learnable GDA in the GCL model has a potential bottleneck in the training, i.e., *the problematic design mechanism of GDA makes the GCL model be updated effectively*. We expect to achieve an advanced GDA that can evolve progressively.



(a) AD-GCL (b) GA2C (ours)

Figure 1: Comparison of AD-GCL and GA2C. The results are from the graph classification task in the BBBP dataset Hu et al. (2020) composed of 2,039 molecular graphs.

In this work, we propose reinforced GDA for GCL, making the augmentation procedure explore an optimal learning path actively. The view learner in GDA is regarded as an agent that interacts with the encoder network, such as graph isomorphism networks (GIN) Xu et al. (2019a), in the training environment. Based on this idea, we design a Graph Advantage Actor-Critic (GA2C) model where the joint use of the Actor submodel and Critic submodel contributes to the progressive evolution of GDA. To illustrate it, we plot the result of GA2C in Figure 1(b). Surprisingly, like a student doing exercises from easy problems to hard problems, we find that the view learner actively explores a learning path corresponding to the order from simple topology structures to complex ones. At the same time, the AUC performance on the test data improves from 65% to 75% and keeps relatively stable after the 32nd epoch.

The main contributions of this work are summarized as follows:

- We propose a reinforced GDA strategy that formulates a Markov Decision Process for GDA.
- Under this strategy, we design a Graph Advantage Actor-Critic (GA2C) model to achieve continuous and learnable GDA. The view evolution in this model significantly promotes GCL.
- On 17 graph datasets covering 5 different types, we verify that GA2C is robust and superior to other graph contrastive learning models on the graph classification task.

2 PRELIMINARIES

Graph contrastive learning. Graph contrastive learning (GCL) is an important type of unsupervised learning on the graph Wu et al. (2021). As shown in Figure 2, GCL typically includes three main procedures: graph data augmentation (GDA), network encoding, and contrastive loss based optimization. Assume an input graph $\mathcal{G}_i$ is sampled from the distribution $\mathbb{P}_{(\mathcal{G})}$ defined over a universal graph set $\mathcal{G}$, two augmented views are obtained by two specific GDA operations $A_{\acute{\omega}}$ and $A_{\grave{\omega}}$ where $\acute{\omega}$ and $\grave{\omega}$ are two augmentation parameters. $A_{\acute{\omega}}$ and $A_{\grave{\omega}}$ are from the distribution $\mathbb{P}_{(A_1,A_2)}$ defined over all possible parameters of two augmentation strategies A_1 and A_2. The two views are then passed into the encoder network $f_{\boldsymbol{\phi}}(\cdot)$ and the projection network $g_{\boldsymbol{\varphi}}(\cdot)$. Finally, the graph embedding is calculated by an optimization process based on a predefined contrastive loss. The definition of graph contrastive learning is as follows:

Definition 1. **(Graph contrastive learning)**. Given an attributed graph $\mathcal{G}_i{=}(\mathcal{A}_{\mathcal{G}_i}, \mathcal{X}_{\mathcal{G}_i})$ and its two augmented views $\acute{\mathcal{G}}_i{=}(\mathcal{A}_{\acute{\mathcal{G}}_i}, \mathcal{X}_{\acute{\mathcal{G}}_i})$, $\grave{\mathcal{G}}_i{=}(\mathcal{A}_{\grave{\mathcal{G}}_i}, \mathcal{X}_{\grave{\mathcal{G}}_i})$ (adjacency matrices $\mathcal{A}_{\mathcal{G}_i}, \mathcal{A}_{\acute{\mathcal{G}}_i}, \mathcal{A}_{\grave{\mathcal{G}}_i} {\in} \mathbb{R}^{N_i \times N_i}$, attribute matrices $\mathcal{X}_{\mathcal{G}_i}, \mathcal{X}_{\acute{\mathcal{G}}_i}, \mathcal{X}_{\grave{\mathcal{G}}_i} {\in} \mathbb{R}^{N_i \times M_i}$), the goal of graph contrastive learning is to learn a function F_{GCL}: $\mathbb{G} \to \mathbb{Z}$ or $\mathbb{N} \to \mathbb{H}$, where $\mathbb{G}$ is graph space ($\mathcal{G}_i, \acute{\mathcal{G}}_i, \grave{\mathcal{G}}_i {\in} \mathbb{G}$), $\mathbb{N}$ node space ($u, v, \cdots {\in} \mathbb{N}$), $\mathbb{Z}$ graph embedding space, and $\mathbb{H}$ node embedding space. For the graph embedding $\boldsymbol{Z}_i {\in} \mathbb{R}^D$ ($D {\ll} M_i$) in $\mathbb{Z}$, we call it the contrastive embedding of graph $\mathcal{G}_i$. For the node embedding $\boldsymbol{h}_u {\in} \mathbb{R}^D$, we call it the contrastive embedding of node u.

Downstream tasks. GCL is generally used as a pretext task which is followed by downstream tasks such as graph classification, node classification, and link prediction. When GCL is complete, the trained parameters $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$ in the encoder network and projection network are used

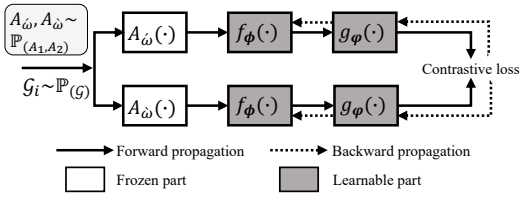


Figure 2: Graph contrastive learning framework.

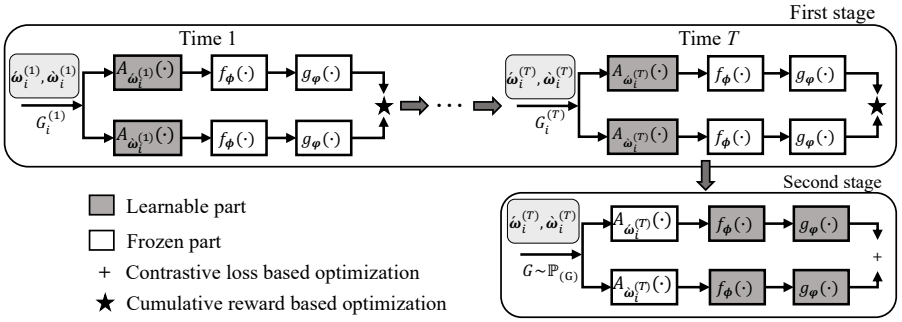


Figure 3: Illustration of the graph reinforced augmentation framework in one training epoch.

as the initial solution of the model in a downstream task. Taking the graph classification task for an example, its definition is as follows:

Definition 2. **(Graph classification)**. Given an input set of n attributed graphs $\mathcal{D} = \{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_n\}$ and the label l_i of each graph $\mathcal{G}_i$, the goal of graph classification is to learn a function $F_{\mathrm{GC}} : \mathbb{G} \to \mathbb{L}$ ($\mathcal{D} \subset \mathbb{G}, l_i \in \mathbb{L}$), where $\mathbb{G}$ is graph space and $\mathbb{L}$ label set.

3 METHODS

In this section, we present the proposed graph reinforced augmentation framework and GA2C model. For frequently used notations in the paper, we list and describe them in Table 1.

3.1 GRAPH REINFORCED AUGMENTATION 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. From this point, reinforcement learning is suitable to characterize an evolving view learner in GDA. We illustrate the graph reinforced augmentation framework in Figure 3. Specifically, there are two novel designs in this framework as follow,

Table 1: Frequently used notations in this paper.

Notations	Descriptions
$\mathcal{D}=\{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_n\}$	Input set of n attributed graphs.
$\mathcal{G}_i=(\mathcal{V}_i, \mathcal{E}_i)$	The i-th attributed graph in $\mathcal{D}$.
$\mathcal{V}_i, \mathcal{E}_i$	Node set and edge set of $\mathcal{G}_i$.
$N_i, M_i \in \mathbb{R}_+$	Node number and attribute number of $\mathcal{G}_i$.
$D \in \mathbb{R}_+$	Hidden size of embeddings.
$\mathcal{A}_{\mathcal{G}_i} \in \mathbb{R}^{N_i \times N_i}$	Adjacency matrix of $\mathcal{G}_i$.
$\mathcal{X}_{\mathcal{G}_i} \in \mathbb{R}^{N_i \times M_i}$	Attribute matrix of $\mathcal{G}_i$.
$\acute{\mathcal{G}}_i=(\mathcal{A}_{\acute{\mathcal{G}}_i}, \mathcal{X}_{\acute{\mathcal{G}}_i})$	First augmented view of $\mathcal{G}_i$.
$\grave{\mathcal{G}}_i=(\mathcal{A}_{\grave{\mathcal{G}}_i}, \mathcal{X}_{\grave{\mathcal{G}}_i})$	Second augmented view of $\mathcal{G}_i$.
$\boldsymbol{Z}_i \in \mathbb{R}^D$	Graph embeddings of $\mathcal{G}_i$.
$\acute{\boldsymbol{Z}}_i^{(t)}, \grave{\boldsymbol{Z}}_i^{(t)} \in \mathbb{R}^D$	Graph embeddings of $\acute{\mathcal{G}}_i, \grave{\mathcal{G}}_i$ at time t.
$\boldsymbol{h}_v^{(k)}$	Node embedding of v at k-th-layer network.
$A_{\acute{\omega}}(\cdot), A_{\grave{\omega}}(\cdot)$	Graph data augmentation (GDA) operations.
$\boldsymbol{\omega}_i^{(t)}$	GDA parameter of $\mathcal{G}_i$ at time t.
$f_{\boldsymbol{\phi}}(\cdot), g_{\boldsymbol{\varphi}}(\cdot)$	Encoder network and projection network.
$\boldsymbol{\theta}_A, \boldsymbol{\theta}_C$	Parameter of Actor, parameter of Critic.
T	Duration time of Actor or Critic.

1. Alternate optimization of contrastive loss and cumulative reward. We design two-stage model learning: the first stage is to maximize the expected cumulative reward between two augmented graph embeddings $\acute{\boldsymbol{Z}}_i^{(t)}$ and $\grave{\boldsymbol{Z}}_i^{(t)}$ at time t by only updating GDA parameters; the second stage is to minimize the InfoNCE loss $\mathcal{L}(\cdot)$ between two views by only updating the encoder and projection parameters $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$. The whole optimization objective is as follows:

$$\min_{\boldsymbol{\phi},\boldsymbol{\varphi}} -\sum_{i=1}^{n} \mathcal{L}(g_{\boldsymbol{\varphi}}(f_{\boldsymbol{\phi}}(A_{\acute{\omega}_i^*}(\mathcal{G}_i))), g_{\boldsymbol{\varphi}}(f_{\boldsymbol{\phi}}(A_{\grave{\omega}_i^*}(\mathcal{G}_i)))), \quad (\acute{\boldsymbol{\omega}}_i^*, \grave{\boldsymbol{\omega}}_i^*) = \arg\max_{\acute{\omega}_i, \grave{\omega}_i} \{\mathcal{J}_A, \mathcal{J}_C\}, \tag{1}$$

where $\mathcal{J}_A$ ($\mathcal{J}_C$) is the cumulative reward function of the Actor (Critic) model.

2. Continuous and learnable GDA. Different from the discrete and frozen GDA in the prior work, the GDA procedure in our framework is continuous and learnable. The continuity ensures that the view learner can adjust the current augmentation parameters according to the state at the previous time. This contributes to that the augmented views evolve to a better result steadily in the training.

3.2 GRAPH ADVANTAGE ACTOR-CRITIC MODEL

In this subsection, we present the design of GA2C in detail, including the Markov decision process for GDA, model architecture, updating Actor, and updating Critic.

Markov decision process for GDA. The determination of GDA parameters is regarded as a Markov decision process: given the present augmented view $\acute{\mathcal{G}}_i^{(t)}$ of graph $\mathcal{G}_i$ at time t, the augmented

view $\acute{\mathcal{G}}_i^{(t+1)}$ of next time is independent of any past view $\acute{\mathcal{G}}_i^{(t')}$ where $t' < t$ (note that we omit the discussion about $\grave{\mathcal{G}}_i^{(t)}$ and $\grave{\boldsymbol{\omega}}_i^{(t+1)}$ to save more space because they have the same property as $\acute{\mathcal{G}}_i^{(t)}$ and $\acute{\boldsymbol{\omega}}_i^{(t+1)}$). It can be formalized as follows:

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

Since edge removing is an effective and popular GDA choice for graph classification, we adopt edge removing as the GDA strategy of GA2C and $\acute{\boldsymbol{\omega}}_i^{(t)}$ represents the removing probability of edges in $\acute{\mathcal{G}}_i$ at time t. A Markov decision process for GDA can be formulated as a 4-tuple (S, A, P, R):

- **State** (S): S is the set of all possible augmented views. A state s in S represents a certain augmented view $\acute{\mathcal{G}}_i$. Also, the state at time t is denoted as $\acute{\mathcal{G}}_i^{(t)}$.
- **Action** (A): A is the set of all possible augmentation operations. An action a in A represents a certain augmentation operation $A_{\acute{\boldsymbol{\omega}}_i}(\cdot)$. Also, the action at time t is denoted as $A_{\acute{\boldsymbol{\omega}}_i^{(t)}}$. For example, given a graph $\mathcal{G}=\{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}=\{v_1, v_2, v_3\}$ and $\mathcal{E}=\{(v_1, v_2), (v_1, v_3)\}$, we define an edge index dictionary $E=\{1{:}(v_1, v_2), 2{:}(v_1, v_3)\}$. Then the action set for the edge removing strategy is $A=\{[0, 0], [0, 1], [1, 0], [1, 1]\}$ where "0" and "1" represents removing and retaining, respectively (e.g., $[0, 1]$ represents removing the 1st edge (v_1, v_2) and retaining the 2nd edge (v_1, v_3)).
- **Transition probability** (P): P is a state transition probability matrix where the entry $p_{ss'}$ corresponds to the probability transiting from state s to state s'.
- **Reward** (R): R is the total reward of the trajectory measuring the reward value generated by a series of states. Also, the reward at time t is denoted as $R^{(t)}$. To decrease the redundant information between two views as much as possible, we set R_t as the negative mutual information between two augmented graph embeddings $\acute{\boldsymbol{Z}}_i^{(t)}$ and $\grave{\boldsymbol{Z}}_i^{(t)}$, i.e., $R^{(t)} = -\mathcal{I}(\acute{\boldsymbol{Z}}_i^{(t)}, \grave{\boldsymbol{Z}}_i^{(t)})$ where $\acute{\boldsymbol{Z}}_i^{(t)} = g_{\boldsymbol{\varphi}}(f_{\boldsymbol{\phi}}(A_{\acute{\boldsymbol{\omega}}_i}(\mathcal{G}_i)))$ and $\grave{\boldsymbol{Z}}_i^{(t)} = g_{\boldsymbol{\varphi}}(f_{\boldsymbol{\phi}}(A_{\grave{\boldsymbol{\omega}}_i}(\mathcal{G}_i)))$.

Model architecture. The Graph Advantage Actor-Critic (GA2C) model contains two sub-models of Actor and Critic. The advantage function in GA2C can be approximated by the temporal difference (TD) error. We illustrate GA2C in Figure 4. The Actor model with parameter $\boldsymbol{\theta}_A$ receives the information in $\acute{\mathcal{G}}_i^{(t)}$ and estimates the GDA parameter $\acute{\boldsymbol{\omega}}_i^{(t+1)}$ corresponding to an augmentation operation $A_{\acute{\boldsymbol{\omega}}_i^{(t+1)}}(\cdot)$ by three-layer graph neural networks:

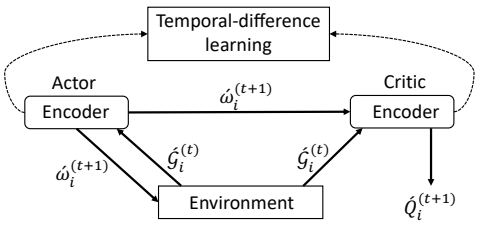


Figure 4: Graph advantage Actor-Critic.

$$\begin{aligned}\acute{\boldsymbol{\omega}}_i^{(t+1)} &= \mathrm{BN}(\mathrm{CONCAT}(\mathrm{READOUT}(\left\{\boldsymbol{h}_v^{(k)}|v\in\acute{\mathcal{G}}_i^{(t)}\right\})|k{=}1,2,3)),\\ \boldsymbol{h}_v^{(k)} &= \mathrm{MLP}_{\boldsymbol{\theta}_A}^{(k)}(\boldsymbol{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_{\acute{\mathcal{G}}_i^{(t)}}(v)} \boldsymbol{h}_u^{(k-1)}),\end{aligned} \tag{3}$$

where $\mathrm{BN}(\cdot)$ is the batch normalization layer, $\mathrm{READOUT}(\cdot)$ the readout layer calculating the graph embedding from node embeddings, $\boldsymbol{h}_v^{(k)}$ the embedding of node v ($v\in\mathcal{V}_i$) at the k-th layer neural network, $\mathcal{N}_{\acute{\mathcal{G}}_i^{(t)}}(v)$ the node v's neighbor set in $\acute{\mathcal{G}}_i^{(t)}$, and $\boldsymbol{h}_u^{(0)}$ corresponds to the attribute vector of node u. The Critic model with parameter $\boldsymbol{\theta}_C$ receives $\acute{\mathcal{G}}_i^{(t)}$, $\acute{\boldsymbol{\omega}}_i^{(t+1)}$ and estimates the Q-value $\acute{\boldsymbol{Q}}_i^{(t+1)}$ by three-layer graph neural networks. Like simulating an action on a state, we conduct the edge removing operation with parameter $\acute{\boldsymbol{\omega}}_i^{(t+1)}$ on $\acute{\mathcal{G}}_i^{(t)}$ and then obtain $\acute{\mathcal{G}}_i^{(t+1)}$:

$$\acute{\mathcal{G}}_i^{(t+1)} = A_{\acute{\boldsymbol{\omega}}_i^{(t+1)}}(\acute{\mathcal{G}}_i^{(t)}). \tag{4}$$

The node v's neighbor set in $\acute{\mathcal{G}}_i^{(t+1)}$ is denoted as $\mathcal{N}_{\acute{\mathcal{G}}_i^{(t+1)}}$. The Critic model is formalized as follows:

$$\begin{aligned}\acute{\boldsymbol{Q}}_i^{(t+1)} &= \mathrm{BN}(\mathrm{CONCAT}(\mathrm{READOUT}(\left\{\boldsymbol{h}_v^{(k)}|v\in\acute{\mathcal{G}}_i^{(t+1)}\right\})|k{=}1,2,3)),\\ \boldsymbol{h}_v^{(k)} &= \mathrm{MLP}_{\boldsymbol{\theta}_C}^{(k)}(\boldsymbol{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_{\acute{\mathcal{G}}_i^{(t+1)}}(v)} \boldsymbol{h}_u^{(k-1)}).\end{aligned} \tag{5}$$

We train the Actor model and Critic model using the gradient ascent algorithm. Specifically, it contains the following two aspects.

Updating Critic. The optimization objective of the Critic model is $\mathcal{J}_C = -\sum_{t=1}^{T}[\acute{\boldsymbol{A}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i)]^2$ and the gradient of the Critic parameter $\boldsymbol{\theta}_C$ is computed as follows:

$$\frac{d}{d\boldsymbol{\theta}_C}\sum_{t=1}^{T}[\acute{\boldsymbol{A}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i)]^2, \tag{6}$$

where the advantage function $\acute{\boldsymbol{A}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i)$ denotes how much better it is to take the action of edge removing with parameter $\acute{\boldsymbol{\omega}}_i$ compared to the average action at the given state $\acute{\mathcal{G}}_i^{(t)}$. For $\acute{\boldsymbol{A}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i)$, we use the difference of the Q-value $\acute{\boldsymbol{Q}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i)$ and V-value $\acute{\boldsymbol{V}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}|\grave{\mathcal{G}}_i^{(t)})$:

$$\acute{\boldsymbol{A}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i) = \acute{\boldsymbol{Q}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i) - \acute{\boldsymbol{V}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}|\grave{\mathcal{G}}_i^{(t)}), \tag{7}$$

where $\acute{\boldsymbol{Q}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i)$ is defined in Eq. 5 and $\acute{\boldsymbol{V}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}|\grave{\mathcal{G}}_i^{(t)})$ is defined as follows:

$$\acute{\boldsymbol{V}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}|\grave{\mathcal{G}}_i^{(t)}) = \sum_{j=1}^{t}\gamma^j R^{(t)} = \sum_{j=1}^{t} -\gamma^j \mathcal{I}(g_{\boldsymbol{\varphi}}(f_{\boldsymbol{\phi}}(A_{\acute{\boldsymbol{\omega}}_i}(\mathcal{G}_i)))), g_{\boldsymbol{\varphi}}(f_{\boldsymbol{\phi}}(A_{\grave{\boldsymbol{\omega}}_i}(\mathcal{G}_i))))). \tag{8}$$

Updating Actor. The optimization objective of Actor is $\mathcal{J}_A = \sum_{t=1}^{T}\log\boldsymbol{\pi}_i(\acute{\boldsymbol{\omega}}_i|\acute{\mathcal{G}}_i^{(t)})\cdot\acute{\boldsymbol{A}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i) + \lambda\mathcal{J}_{reg}$ (λ is the coefficient of regularization term) and the gradient of the Actor parameter $\boldsymbol{\theta}_A$ is computed as follows:

$$\sum_{t=1}^{T}\frac{d\log\boldsymbol{\pi}_i(\acute{\boldsymbol{\omega}}_i|\acute{\mathcal{G}}_i^{(t)})}{d\boldsymbol{\theta}_A}\cdot\acute{\boldsymbol{A}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i) + \frac{\lambda\mathcal{J}_{reg}}{d\boldsymbol{\theta}_A}, \tag{9}$$

where the policy function $\boldsymbol{\pi}_i(\acute{\boldsymbol{\omega}}_i|\acute{\mathcal{G}}_i^{(t)})$ is designed by using the sigmoid function to turn each entry $[\acute{\boldsymbol{\omega}}_i^{(t+1)}]_{j,k}$ in $\acute{\boldsymbol{\omega}}_i^{(t+1)}$ (defined in Eq. 3) into a probability value:

$$[\boldsymbol{\pi}_i(\acute{\boldsymbol{\omega}}_i|\acute{\mathcal{G}}_i^{(t)})]_{j,k} = \frac{1}{1+e^{-[\acute{\boldsymbol{\omega}}_i^{(t+1)}]_{j,k}}} \tag{10}$$

and the regularization term $\mathcal{J}_{reg} = \acute{\boldsymbol{\omega}}_i^{(t+1)}$ prevents GA2C not to remove massive edges from the graph. The pseudocode for the complete training process of GA2C is shown in Algorithm 1. For the computational complexity analysis of GA2C, please see Appendix A.

Algorithm 1: Training process of GA2C

Data: A set of graphs $\mathcal{D} = \{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_n\}$, encoder network $f_{\boldsymbol{\phi}}$, projection network $g_{\boldsymbol{\varphi}}$, Actor with parameter $\boldsymbol{\theta}_A$, Critic with parameter $\boldsymbol{\theta}_C$, epoch number N_{epoch}, duration time T, learning rate α_A of Actor, learning rate α_C of Critic, and learning rate α_E of encoder or projection network.

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

Initialize $\boldsymbol{\phi}$, $\boldsymbol{\varphi}$, $\boldsymbol{\theta}_A$, and $\boldsymbol{\theta}_C$ using the Glorot uniform initializer; Initialize k: $k \leftarrow 0$;
while $k < N_{epoch}$ **do**
  for $\mathcal{G}_i \in \mathcal{D}$ **do**
    Initialize t: $t \leftarrow 0$; Initialize $\acute{\mathcal{G}}_i^{(t)}$: $\acute{\mathcal{G}}_i^{(t)} \leftarrow \mathcal{G}_i$; Initialize $\acute{\boldsymbol{\omega}}_i^{(t)}$ as an all-one vector;
    while $t < T$ **do**
      Calculate $\acute{\boldsymbol{\omega}}_i^{(t+1)}$ based on Eq. 3 and take action based on Eq. 4;
      Calculate $\boldsymbol{\pi}_i(\acute{\boldsymbol{\omega}}_i|\acute{\mathcal{G}}_i^{(t)})$ based on Eq. 10;
      Calculate $\acute{\boldsymbol{A}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i)$ based on Eqs. 5,7,8; $t \leftarrow t+1$;
    end
    Update $\boldsymbol{\theta}_A$: $\boldsymbol{\theta}_A \leftarrow \boldsymbol{\theta}_A + \alpha_A\sum_{t=1}^{T}\frac{d\log\boldsymbol{\pi}_i(\acute{\boldsymbol{\omega}}_i|\acute{\mathcal{G}}_i^{(t)})}{d\boldsymbol{\theta}_A}\cdot\acute{\boldsymbol{A}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i) + \frac{\lambda\mathcal{J}_{reg}}{d\boldsymbol{\theta}_A}$;
    Update $\boldsymbol{\theta}_C$: $\boldsymbol{\theta}_C \leftarrow \boldsymbol{\theta}_C + \alpha_C\frac{d}{d\boldsymbol{\theta}_C}\sum_{t=1}^{T}[\acute{\boldsymbol{A}}_i^{(t)}(\acute{\mathcal{G}}_i^{(t)}, \acute{\boldsymbol{\omega}}_i)]^2$;
    Calculate the InfoNCE loss $\mathcal{L}$ with $\acute{\boldsymbol{\omega}}_i^* = \acute{\boldsymbol{\omega}}_i^{(T)}$ based on Eq. 1;
    Update $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$: $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \alpha_E\frac{d\mathcal{L}}{d\boldsymbol{\phi}}$, $\boldsymbol{\varphi} = \boldsymbol{\varphi} - \alpha_E\frac{d\mathcal{L}}{d\boldsymbol{\varphi}}$;
  end
  $k \leftarrow k+1$;
end

4 EXPERIMENTS

In the experiments, we first describe the used datasets and baselines (for the detailed experimental setups, please see Appendix B). Next, we present the experimental results on three downstream tasks. Then we conduct an ablation study and sensitivity analysis on GA2C. Finally, we analyze the cumulative reward in GA2C and visualize the embedding results learned by GA2C.

4.1 DATASETS AND BASELINES

Datasets. We adopt the same datasets and data splitting in AD-GCL Suresh et al. (2021). These datasets are for graph classification and from TU Benchmark Morris et al. (2020) and Open Graph Benchmark (OGB) Hu et al. (2020). They cover five types of graphs including biochemical molecules (NCI1, PROTEINS, MUTAG, and DD), social networks (COLLAB, RDT-B, RDT-M, IMDB-B, and IMDB-M), physical chemistry (ESOL, Lipophilicity, and FreeSolv), biophysics (BACE), and physiology (BBBP, ClinTox, Tox21, and SIDER). The statistics of these datasets are shown in Table 2. For the datasets containing more than one task (e.g., ClinTox has two binary classification tasks for 1,477 drug compounds with known chemical structures), we report the mean metric values over all tasks. For the datasets of node classification and link prediction, please see Appendix C.

Baselines. For graph classification, we compare GA2C with eight baselines totally, including one supervised learning model (F-GIN Xu et al. (2019b)), three GCL models with frozen GDA (RU-GIN Xu et al. (2019b), InfoGraph Sun et al. (2020), and GraphCL You et al. (2020)), three GCL models with learnable GDA (AD-GCL Suresh et al. (2021), LP-InfoMin You et al. (2022), and LP-InfoBN You et al. (2022)), and one GCL model with no GDA (SimGRACE Xia et al. (2022)). For node classification, we compare GA2C with five baselines including four GZL models with frozen GDA (DGI Velickovic et al. (2019), MVGRL Hassani & Khasahmadi (2020), GraphCL You et al. (2020), and BGRL Thakoor et al. (2021)) and one GZL model with precomputed GDA (GCA Zhu et al. (2021)). For link prediction, we compare GA2C with four GCL models (DGI Velickovic et al. (2019), MVGRL Hassani & Khasahmadi (2020), GMI Peng et al. (2020), and SAIL Yu et al. (2022b)) and one unsupervised model (CAN Meng et al. (2019)) which are from Yu et al. (2022b).

4.2 PERFORMANCE ON DOWNSTREAM TASKS

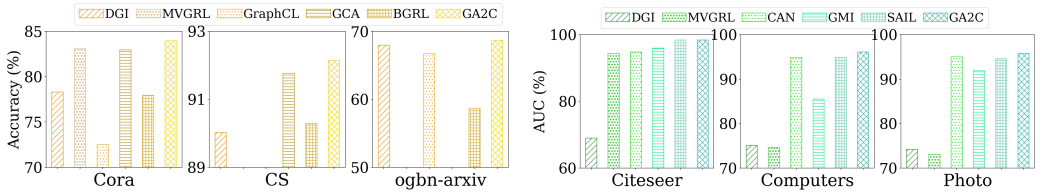
Graph classification. We run experiments on five types of graphs and the corresponding experimental results are shown in Table 3. From Table 3, we can see that the proposed model GA2C has better overall performance than other baselines on biochemistry and social networks. On some datasets (e.g., PROTEINS, DD, and RDT-M), GA2C has even outperformed the supervised F-GIN model. Particularly, on IMDB-M, GA2C relatively outperforms F-GIN by about 4.1%. Based on this observation, we guess that the reinforced GDA plays an important role in GCL. From the RMSE (root mean square error) or AUC results on physical chemistry, biophysics, and physiology, we can see that GA2C still has good performance. For example, on the datasets of BACE and BBBP, GA2C relatively outperforms the runner-ups AD-GCL and LP-InfoBN by 4.03% and 6.57%, respectively. Moreover, on ESOL, FreeSolv, and Tox21, seven baselines fall behind the supervised F-GIN but GA2C outperforms it. This indicates the advantage of reinforced GDA in GCL again.

Table 2: Statistics of the used datasets. The left part is the biochemical and social networks from TU Benchmark Morris et al. (2020). The right part is the physical chemistry, biophysics, and physiology from Open Graph Benchmark (OGB) Hu et al. (2020). Here "#Nodes" and "#Edges" are the average node number and average edge number in each graph, respectively.

Dataset	Type	#Graph	#Node	#Edge	#Class	Dataset	Type	#Graph	#Node	#Edge	#Task
NCI1	Biochemistry	4,110	29.87	32.30	2	ESOL	Physical chemistry	1,128	13.3	13.7	1
PROTEINS	Biochemistry	1,113	39.06	72.82	2	Lipophilicity	Physical chemistry	4,200	27.0	29.5	1
MUTAG	Biochemistry	188	17.93	19.79	2	FreeSolv	Physical chemistry	642	8.7	8.4	1
DD	Biochemistry	1,178	284.32	715.66	2	BACE	Biophysics	1,513	34.1	36.9	1
COLLAB	Social networks	5,000	74.5	2457.78	3	BBBP	Physiology	2,039	24.1	26.0	1
RDT-B	Social networks	2,000	429.6	497.75	2	ClinTox	Physiology	1,477	26.2	27.9	2
RDT-M	Social networks	4,999	508.8	594.87	5	Tox21	Physiology	7,831	18.6	19.3	12
IMDB-B	Social networks	1,000	19.8	96.53	2	SIDER	Physiology	1,427	33.6	35.4	27
IMDB-M	Social networks	1,500	13.0	65.94	3						

Table 3: Comparison results of baselines and GA2C on TU Benchmark and OGB Benchmark. For each column, apart from the result of F-GIN (a supervised learning model, denoted as "sup."), the best result is bolded and the runner-up is underlined. "Lipo" denotes the dataset of Lipophilicity.

Method	AUC (%)(↑)					Accuracy (%)(↑)									RMSE (↓)		
	BACE	BBBP	ClinTox	Tox21	SIDER	NCI1	PROTEINS	MUTAG	DD	COLLAB	RDT-B	RDT-M	IMDB-B	IMDB-M	ESOL	Lipo	FreeSolv
F-GIN (sup.)	72.97	68.17	88.14	74.91	57.60	78.27	72.39	90.41	74.87	74.82	86.79	53.28	71.83	48.46	1.173	0.757	2.755
RU-GIN	75.07	64.48	72.29	71.53	62.29	62.98	69.03	87.61	74.22	63.08	58.97	27.52	51.86	32.81	1.706	1.075	7.526
InfoGraph	74.74	66.33	64.50	69.74	60.54	68.13	72.57	87.71	75.23	70.35	78.79	51.11	71.11	48.66	1.344	1.005	10.01
GraphCL	74.32	68.22	74.92	72.40	61.76	68.54	72.86	88.29	74.70	71.26	82.63	53.05	70.80	48.49	1.272	0.910	7.679
AD-GCL	76.37	68.24	80.77	71.42	63.19	69.67	73.59	89.25	74.49	73.32	85.52	53.00	71.57	49.04	1.217	0.842	5.150
LP-InfoMin	75.82	70.32	71.83	72.52	60.62	69.53	71.90	88.69	74.07	71.52	81.27	53.05	71.08	48.79	1.245	0.974	7.698
LP-InfoBN	76.23	70.45	76.38	72.10	60.95	69.08	71.34	88.68	73.50	72.43	80.84	53.12	70.25	48.43	1.220	0.955	7.304
SimGRACE	76.05	70.24	76.70	71.39	60.61	68.62	72.15	89.15	74.40	71.66	81.59	53.04	71.31	49.56	1.387	0.920	5.485
GA2C (ours)	**79.45**	**75.08**	**81.62**	**75.41**	**63.84**	**72.34**	**73.77**	**90.11**	**75.64**	**73.50**	**85.80**	**53.95**	**72.60**	**50.47**	**1.034**	**0.833**	**2.540**



(a) Node classification (b) Link prediction

Figure 5: Comparison results on the tasks of node classification and link prediction.

Node classification. The results of the node classification task are shown in Figure 5(a). If the out-of-memory error happens in some model (e.g., GCA on ogbn-arxiv), we do not plot its corresponding bar. From Figure 5(a), we can see that GA2C and GCA (precomputed method) perform better than other baselines on Cora and CS. We infer that precomputed GDA and learnable GDA can generate more suitable views for the input graph than the trial-and-error method. In addition, GA2C outperforms other baselines in the three datasets, which indicates the graph reinforced augmentation framework works well on the node classification task.

Link prediction. The results of the node classification task are shown in Figure 5(b). From it, we can see that GA2C is competitive with the SOTA GCL models. Also, SAIL adopts self-augmented views to advance the expressivity of GNNs and achieves the runner-up in the comparison. Therefore, we conclude that the procedure of GDA is crucial for either GCL or GNNs.

4.3 Ablation Study

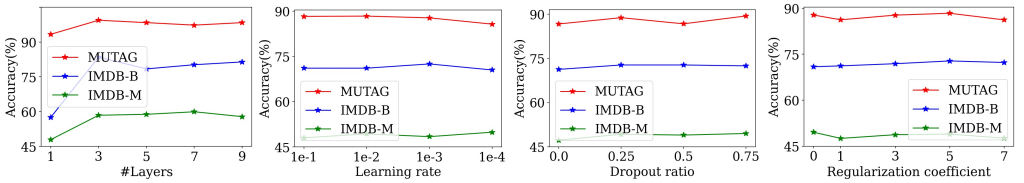
To evaluate which reinforcement learning model is more suitable for the graph reinforced augmentation framework, we conduct an ablation study on GA2C by replacing the advantage Actor-Critic model with other two reinforce models: vanilla policy gradient (namely GPG) and standard Actor-Critic model (namely GAC). The experimental results are shown in Table 4. From it, we can see the obvious advantage of GA2C over GPG and GAC. We explain that GA2C using advantage Actor-Critic has more stable performance and lower variance, which improves GPG and GAC. In addition, we evaluate the effects of different GDA strategies on the model performance on downstream tasks. The used GDA strategies include edge removing ("ER"), edge perturbation ("EP"), and attribute masking ("AM"). The corresponding results are shown in the last three lines in Table 4. We find that AM fails behind the EP and ER. AM is not suitable for GA2C because the evolution of topological structure, rather than semantic information, reflects the significant changes in graph characteristics.

4.4 Sensitivity Analysis

In this part, we conduct the sensitivity analysis to GA2C. The hyperparameters to be analyzed include the number of encoder network layers, learning rate α_E (note that we set $\alpha_E = \alpha_A = \alpha_C$), dropout ratio, and regularization coefficient λ. Based on the testing results on three representative datasets (i.e., MUTAG, IMDB-B, and IMDB-M) under different hyperparameter values, we plot the Accuracy curve in Figure 6. From it, we can see that GA2C is robust to these hyperparameters apart from the number of encoder network layers. Based on the results in Figure 6(a), we find that the optimal number of encoder network layers is 3 from the candidate set of $\{1, 3, 5, 7, 9\}$. When the number

Table 4: The results of ablation study. "ER", "ED", and "AM" denote edge removing, edge perturbation, and attribute masking, respectively.

model	AUC (%)(↑)					Accuracy (%)(↑)									RMSE(↓)		
	BACE	BBBP	ClinTox	Tox21	SIDER	NCI1	PROTEINS	MUTAG	DD	COLLAB	RDT-B	RDT-M	IMDB-B	IMDB-M	ESOL	Lipo	FreeSolv
GPG+ER	**79.69**	74.94	79.45	74.93	63.52	71.24	71.33	87.78	75.38	72.72	84.60	53.43	71.20	49.87	1.056	0.856	2.671
GAC+ER	78.42	74.16	77.41	75.02	63.41	71.58	73.04	89.97	75.04	72.36	83.55	**54.05**	70.90	50.07	1.054	0.873	2.698
GA2C+ER	79.45	**75.08**	**81.62**	**75.41**	**63.84**	**72.34**	**73.77**	**90.11**	**75.64**	**73.50**	**85.80**	53.95	**72.60**	50.47	**1.034**	**0.833**	**2.540**
GA2C+EP	78.50	74.06	79.30	74.35	61.21	72.30	73.43	89.03	75.12	73.45	84.55	53.95	72.10	**50.90**	1.062	0.859	2.662
GA2C+AM	70.34	72.38	70.55	71.79	56.84	66.40	70.05	86.46	73.37	70.09	81.62	51.18	69.95	49.65	1.088	0.865	2.671



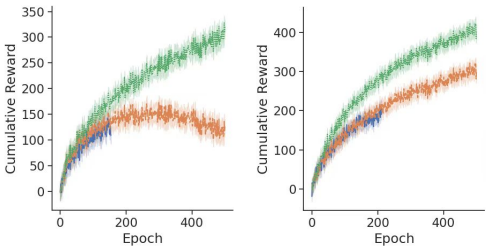
(a) Number of encoder network layers (b) Learning rate α_E (c) Dropout ratio (d) Regularization coefficient λ

Figure 6: The results of sensitivity analysis on MUTAG, IMDB-B, and IMDB-M.

is set too small, GA2C has bad performance because shallow layers are not enough to encode the abundant structural and semantic information in the graph.

4.5 Analysis of Cumulative Reward

In this part, we evaluate the cumulative reward of GPG, GAC, and GA2C in the training phase. The learning curves in terms of cumulative reward on two representative datasets (i.e., ESOL and FreeSolv) are shown in Figure 7. From Figure 7, we can see that the performance of GPG (in blue) is unstable and worse than the other two models due to the existence of exploding gradients. Specifically, the exploding gradients happen at the 156th epoch and 213-rd epoch on ESOL and FreeSolv, respectively. By replacing the sampled reward values with the expected reward values learned by the Critic, both GAC (in red) and GA2C (in green) keep stable performance in the training.

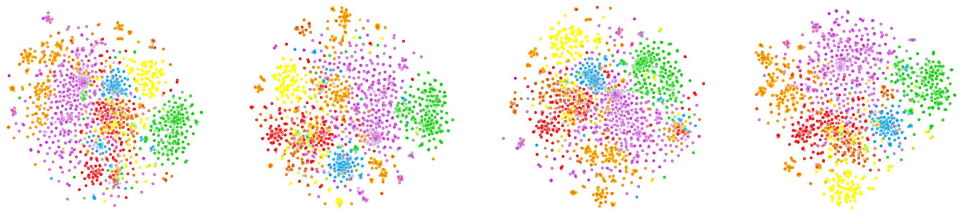


(a) Results on ESOL (b) Results on FreeSolv

Figure 7: The learning curves in terms of cumulative reward on ESOL and FreeSolv. We run all models five times respectively. The solid line represents the mean value of cumulative reward and the shadowed area is enclosed by the min and max value of five training runs.

4.6 Visualization

To evaluate the quality of node embeddings learned by baselines and GA2C qualitatively, we plot their t-SNE 2D projection results on Cora in Figure 8. We select DGI, GCA, and MVGRL as representatives because they perform well on Cora (see Figure 5(a)). From Figure 8(d), we can see that the boundary of clusters is clear, which indicates the embedding learned by GA2C is reasonable. On the contrary, we notice that the boundary of clusters in Figure 8(a) is blurred, which reveals that the embeddings learned by DGI are not well matched to their own clusters.



(a) DGI (b) GCA (c) MVGRL (d) GA2C

Figure 8: Embedding visualization results on Cora. Different colors represent different classes.

5 RELATED WORK

5.1 GRAPH CONTRASTIVE LEARNING

GCL has three procedures: graph data augmentation (GDA), network encoding, and contrastive loss based optimization. According to the difference in GDA, we categorize GCL methods as follows:

GCL Methods with frozen graph data augmentation. Traditional GCL methods You et al. (2020); Sun et al. (2020) adopt frozen augmentation parameters to generate different graph views. Generally, the augmentation parameters are set empirically (i.e., trial-and-error). Different from the trial-and-error method, GCA Zhu et al. (2021) adopts precomputed method: it calculates GDA parameters based on some indicators of centrality in the graph (e.g., degree centrality and eigenvector centrality) and uses these precomputed parameters to augment the graph.

GCL Methods with learnable graph data augmentation. For the GCL with frozen augmentation, there is a risk that the redundant information is captured by the InfoMax principle Tschannen et al. (2020). The redundant information hinders the better performance of GZL on the downstream task. To avoid this, learnable GDA is introduced into GCL, i.e., the parameters of GDA can be 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, You et al. (2022) extend the frozen and discrete GDA parameters to the learnable ones and leverage both principles of information minimization (InfoMin) and information bottleneck (InfoBN) to regularize the learned GDA parameters.

GCL Methods with no graph data augmentation. Recent work removes GDA from GCL and adopts the learning framework without GDA to extract representations from graphs. For example, SimGRACE Xia et al. (2022) takes the raw graph as the input and uses the Graph Neural Networks (GNNs) encoder as well as its perturbed version to generate two correlated views for contrast. Instead of GDA, AF-GCL Wang et al. (2022) constructs positive and negative pairs based on the representations obtained by GNNs. SimGCL Yu et al. (2022a) creates contrastive views by adding random noises to the representations obtained by GNNs.

Compared to prior work, our work differs in that we introduce reinforced GDA into GCL, making the view learner explore an optimal learning path actively.

5.2 GRAPH REINFORCEMENT LEARNING

To address the problem that existing GNNs ignore the semantics of subgraphs, SUGAR Sun et al. (2021) learns the representations of the significant subgraphs which are adaptively selected by Q-learning. Policy-GNN Lai et al. (2020) adopts the DQN algorithm to determine the aggregation range of nodes in a large-scale graph. To adaptively learn the optimal curvature in the graph, ACE-HGNN Fu et al. (2021) leverages multi-agent reinforcement learning where the agents are updated by the Nash Q-leaning algorithm. Removing noise data from graphs is helpful to improve the performance of graph representation learning. GDPNet Xue et al. (2021) and GAM Lee et al. (2018) adopt Markov Decision Process (MDP) and Partially Observable Markov Decision Process (POMDP) to model the noisy neighborhoods removing process and the information collection process, respectively. For heterogeneous graphs, RIOGNN Peng et al. (2022) leverages the reinforcement learning algorithm to select the most similar neighbors within a relation; RL-HGNN Zhong et al. (2020) models the process of designing meta-paths as MDP to replace the manual designing process.

Different from these graph reinforcement learning methods, our work uses the Advantage Actor-Critic model to adjust the current optimal GDA parameters adaptively.

6 CONCLUSION

In this paper, we design a novel graph reinforced augmentation framework which can ensure that the augmented views evolve well to promote graph contrastive learning. Under this framework, the graph advantage Actor-Critic (GA2C) model is proposed to learn graph embeddings in an unsupervised manner. Through extensive experiments, we verify that GA2C outperforms the SOTA graph contrastive learning models on downstream tasks such as graph classification, node classification, and link prediction. In the future, we plan to study graph contrastive learning with learnable graph data augmentation on heterogeneous graphs and text-rich graphs.

REFERENCES

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 M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 9990–9999. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/52aaa62e71f829d41d74892a18a11d59-Paper.pdf.

Micah D Carroll, Anca Dragan, Stuart Russell, and Dylan Hadfield-Menell. Estimating and penalizing induced preference shifts in recommender systems. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2686–2708. PMLR, 17–23 Jul 2022.

Xingcheng Fu, Jianxin Li, Jia Wu, Qingyun Sun, Cheng Ji, Senzhang Wang, Jiajun Tan, Hao Peng, and Philip S. Yu. ACE-HGNN: adaptive curvature exploration hyperbolic graph neural network. In James Bailey, Pauli Miettinen, Yun Sing Koh, Dacheng Tao, and Xindong Wu (eds.), *IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7-10, 2021*, pp. 111–120. IEEE, 2021.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pp. 4116–4126. PMLR, 2020.

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.

Kwei-Herng Lai, Daochen Zha, Kaixiong Zhou, and Xia Hu. Policy-gnn: Aggregation optimization for graph neural networks. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (eds.), *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pp. 461–471. ACM, 2020.

John Boaz Lee, Ryan A. Rossi, and Xiangnan Kong. Graph classification using structural attention. In Yike Guo and Faisal Farooq (eds.), *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 1666–1674. ACM, 2018.

Antonis Matakos, Çigdem Aslay, Esther Galbrun, and Aristides Gionis. Maximizing the diversity of exposure in a social network. *IEEE Trans. Knowl. Data Eng.*, 34(9):4357–4370, 2022.

Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. Co-embedding attributed networks. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pp. 393–401, 2019.

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 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL www.graphlearning.io.

Hao Peng, Ruitong Zhang, Yingtong Dou, Renyu Yang, Jingyi Zhang, and Philip S. Yu. Reinforced neighborhood selection guided multi-relational graph neural networks. *ACM Trans. Inf. Syst.*, 40 (4):69:1–69:46, 2022.

Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, pp. 259–270, 2020.

Chih-Ya Shen, De-Nian Yang, Wang-Chien Lee, and Ming-Syan Chen. Activity organization for friend-making optimization in online social networks. *IEEE Trans. Knowl. Data Eng.*, 34(1): 122–137, 2022.

Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1lfF2NYvH.

Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Yuanxing Ning, Philip S. Yu, and Lifang He. SUGAR: subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. In Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (eds.), *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pp. 2081–2091. ACM / IW3C2, 2021.

Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. Bootstrapped representation learning on graphs. *ICLR 2021 Workshop*, 2021.

Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkxoh24FPH.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

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

Yunhao Wang, Yue Zhao, Audrey Bollas, Yuru Wang, and Kin Fai Au. Nanopore sequencing technology, bioinformatics and applications. *Nature biotechnology*, 39(11):1348–1365, 2021.

Lirong Wu, Haitao Lin, Zhangyang Gao, Cheng Tan, Stan Li, et al. Self-supervised on graphs: Contrastive, generative, or predictive. *arXiv preprint arXiv:2105.07342*, 2021.

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 ACM Web Conference 2022*, pp. 1070–1079, 2022.

Da Xu, Yuting Ye, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. From intervention to domain transportation: A novel perspective to optimize recommendation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=jT1EwXu-4hj.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=ryGs6iA5Km.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=ryGs6iA5Km.

Fuzhao Xue, Aixin Sun, Hao Zhang, and Eng Siong Chng. Gdpnet: Refining latent multi-view graph for relation extraction. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 14194–14202. AAAI Press, 2021.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33: 5812–5823, 2020.

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*, pp. 1300–1309, 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*, pp. 1294–1303, 2022a.

Lu Yu, Shichao Pei, Lizhong Ding, Jun Zhou, Longfei Li, Chuxu Zhang, and Xiangliang Zhang. Sail: Self-augmented graph contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8927–8935, 2022b.

Zhiqiang Zhong, Cheng-Te Li, and Jun Pang. Reinforcement learning enhanced heterogeneous graph neural network. *CoRR*, abs/2010.13735, 2020.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. In *GRL+@ICML 2020*, 2020.

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*, pp. 2069–2080, 2021.

A COMPUTATIONAL COMPLEXITY ANALYSIS

Let L_E be the encoder network layer number, L_A neural network layer number in Actor (or Critic), $N_{avg.}$ average node number in each augmented graph, and $M_{avg.}$ average attribute number in each augmented graph, for each training epoch, the total computational complexity of GA2C is O($n(TL_A + L_E)N_{avg.}M_{avg.}^2$)) where n and T are the graph number and duration time of Actor or Critic.

B EXPERIMENTAL SETUPS AND METRICS

Experimental setups. We use Adam optimizer to train the encoder network, projection network, Actor model, and Critic model in GA2C. The learning rates α_E, α_A, and α_C are set as 0.01 for COLLAB, RDT-M and 0.001 for other datasets. The layer number of the encoder network is set as 3. For biochemistry and social networks, the embedding dimension is set as 32 and the batch size is set as 128; for physical chemistry, biophysics, and physiology, the embedding dimension is set as 300 and the batch size is set as 64. The drop ratio and regularization coefficient are set as 0.0 and 5.0, respectively.

For the downstream tasks, we adopt Linear Support Vector Classification (LinearSVC) as the downstream classifier. We follow the data splitting method in AD-GCL Suresh et al. (2021): for OGB Benchmark, we adopt the proceed data including the training set, validation set, and test set; for TU Benchmark, we adopt 10-fold cross-validation where the total data are split as 70% training set, 20% validation set, and 10% test set.

Metrics. For the regression task (i.e., ESOL, Lipophilicity, and FreeSolv), we adopt the metric of RMSE (root mean square error). For the classification task, we evaluate the dataset of biochemical and social networks under the metric of Accuracy and the other datasets under the metric of AUC (area under the receiver operating characteristic curve).

C DATASETS FOR NODE CLASSIFICATION AND LINK PREDICTION

Datasets for node classification. We use three datasets for the node classification task, including Cora, CS, and ogbn-arxiv. Cora is a citation network where the node and edge represent the paper and citation relationship between papers, respectively. The node attributes represent the paper topics. For Cora, the task is to classify each paper into a domain class. CS is a collaboration network where the node and edge represent the author and co-author relationship between authors, respectively. The node attributes represent the keyword information in the papers published by the author. For CS, the task is to classify each author into a research field. ogbn-arxiv is also a citation network where the papers are all from the open-access repository of arXiv. The detailed statistics of the above three datasets are shown in Table 5.

Table 5: The statistics of the used datasets for the node classification task. The last column is the ratio of node numbers in the training set, validation set, and test set.

Dataset	#Node	#Edge	#Feature	#Class	Split ratio
Cora	2,708	5,429	1,433	7	1:1:8
CS	18,333	81,894	6,805	15	1:1:8
ogbn-arxiv	169,343	1,166,243	128	40	1:1:8

Datasets for link prediction. We use three datasets for the link prediction task, including Citeseer, Computers, and Photo. The dataset of CiteSeer is a citation network, which consists of 3312 scientific publications classified into one of 6 classes. CiteSeer consists of 4732 links. Each publication in CiteSeer is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3703 unique words. The datasets of Computers and Photo are both co-purchase networks extracted from Amazon, where nodes represent products, edges represent the co-purchased relations of products, and features are bag-of-words vectors extracted from product reviews.

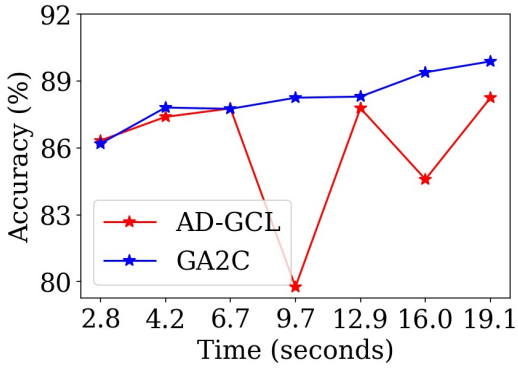


Figure 9: Time-performance curve.

D EXPERIMENTAL RESULTS ON RUNNING TIME

Here we compare the running time of GA2C and baselines when the space complexity of models is the same. On some representative datasets (MUTAG, PROTEINS, and DD), the results are shown in Table 6. We find that GA2C is more efficient than AD-GCL and performs better than all the baselines.

Table 6: The comparison of baselines and GA2C on the running time.

Method	**Time (seconds)**		
	MUTAG	**PROTEINS**	**DD**
GraphCL	150.3	662.6	696.0
AD-GCL	87.3	419.8	601.7
LP-InfoMin	442.3	1948.7	2004.1
LP-InfoBN	440.8	1957.0	2018.4
SimGRACE	<u>68.4</u>	<u>274.6</u>	<u>312.9</u>
GA2C	**65.3**	**266.2**	**310.6**

In adiition, we plot the wall-clock time of the training dynamics of different methods in Figure 9 to observe the relationship between running time and model performance. In this figure, the x-axis represents training time and y-axis represents validation performance. We find that GA2C performs better than AD-GCL in most cases. Also, the performance of AD-GCL is unstable and it has high variance, while GA2C is more stable than AD-GCL.