Graph Self-supervised Learning with Accurate Discrepancy Learning

Dongki Kim[1,*], **Jinheon Baek**[1*], **Sung Ju Hwang**[1,2]
KAIST[1], AITRICS[2], South Korea
cleverki@kaist.ac.kr, jinheon.baek@kaist.ac.kr, sjhwang82@kaist.ac.kr

Abstract

Self-supervised learning of graph neural networks (GNNs) aims to learn an accurate representation of the graphs in an unsupervised manner, to obtain transferable representations of them for diverse downstream tasks. Predictive learning and contrastive learning are the two most prevalent approaches for graph self-supervised learning. However, they have their own drawbacks. While the predictive learning methods can learn the contextual relationships between neighboring nodes and edges, they cannot learn global graph-level similarities. Contrastive learning, while it can learn global graph-level similarities, its objective to maximize the similarity between two differently perturbed graphs may result in representations that cannot discriminate two similar graphs with different properties. To tackle such limitations, we propose a framework that aims to learn the exact discrepancy between the original and the perturbed graphs, coined as *Discrepancy-based Self-supervised LeArning* (D-SLA). Specifically, we create multiple perturbations of the given graph with varying degrees of similarity, and train the model to predict whether each graph is the original graph or the perturbed one. Moreover, we further aim to accurately capture the amount of discrepancy for each perturbed graph using the graph edit distance. We validate our D-SLA on various graph-related downstream tasks, including molecular property prediction, protein function prediction, and link prediction tasks, on which ours largely outperforms relevant baselines[1].

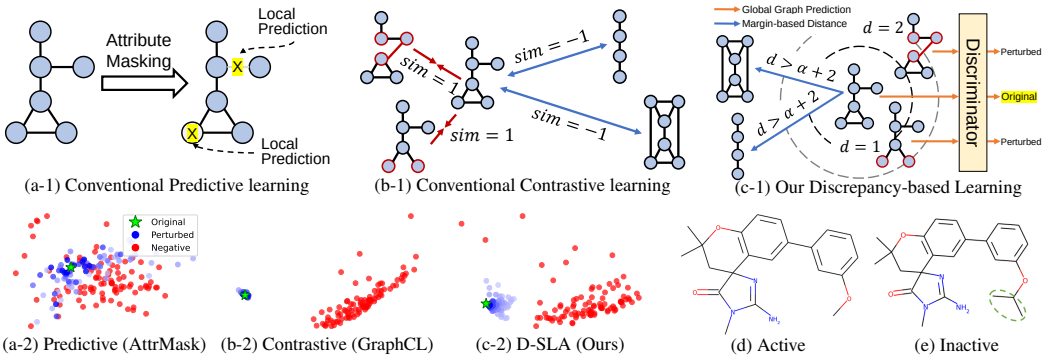
1 Introduction

A graph, consisting of nodes and edges, is a data structure that defines a relationship among objects. Recently, graph neural networks (GNNs) [17, 9, 35, 43], which aim to represent this structure with neural networks, have achieved great successes in modeling real-world graphs such as social networks [5], knowledge graphs [1], biological networks [22], and molecular graphs [41]. However, it is extremely costly and time-consuming to annotate every label of the graphs. For example, labeling the properties of molecular graphs requires time-consuming laboratory experiments [12].

Recently, various self-supervised learning methods for GNNs [12, 51, 13, 31] have been studied to overcome this issue of the lack of labeled graph data. The existing self-supervised learning methods can be classified into two categories: predictive learning and contrastive learning. Predictive learning methods [12, 25, 16] aim to learn representations by predicting contexts of a graph (Figure 1, (a-1)). However, predictive learning schemes are limited as they consider only the subgraphical semantics. Contrastive learning [36, 30, 47, 52, 48, 42], which is a popular self-supervised learning approach in the computer vision field [27, 39, 4, 10], can learn the global semantics by maximizing the similarity between instances (images) perturbed by semantic-invariant perturbations (e.g. color jittering or flipping). Graph contrastive learning methods also use a similar strategy, generating positive examples

[*]Equal Contribution.
[1]Code is available at https://github.com/DongkiKim95/D-SLA

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



(a-1) Conventional Predictive learning (b-1) Conventional Contrastive learning (c-1) Our Discrepancy-based Learning

(a-2) Predictive (AttrMask) (b-2) Contrastive (GraphCL) (c-2) D-SLA (Ours) (d) Active (e) Inactive

Figure 1: **(a) Conventional predictive learning** that aims to predict local attributes by masking them. **(b) Conventional contrastive learning** that could maximize the similarity of dissimilar graphs perturbed from original graphs. **(c) Our discrepancy-based learning** that discriminates an original graph from perturbed graphs by capturing global semantics of them unlike predictive learning, but also that reflects the amount of discrepancy across original, perturbed, and different graphs, unlike contrastive learning. **(a,b,c-2) Graph embedding visualization** trained by each self-supervised learning scheme. **(d),(e) Motivation:** A pitfall of contrastive learning methods, which assumes two similar graphs as the same, despite of their different properties.

with edge perturbation, attribute masking, and subgraph sampling, and then maximizing the similarity between two differently perturbed graphs from the original graph (See Figure 1, (b-1)).

However, is it always correct to assume that the perturbed graphs by such perturbations will preserve the semantics of the original graph? In computer vision, there exists a set of well-defined perturbations, that does not change the human perception of the given image. However, in contrast to images that reside in a continuous domain, graphs are discrete data structures by nature, and thus their properties could become completely different even with slight perturbations. For example, two molecules in Figure 1 (d) and (e) show that, although they have a highly correlated structure, one molecular graph could be actively bound to a set of inhibitors of human β-secretase 1 [38], whereas the other is not. In this case, existing contrastive learning with perturbations may lead the representations for two semantically *dissimilar* graphs to collapse into the one representation (See Figure 1 (b-2)).

To tackle this problem, we propose a novel self-supervised learning method which aims to learn the *discrepancy* between graphs, which we refer to as **D**iscrepancy-based **S**elf-supervised **L**e**A**rning (D-SLA). Specifically, we first perturb the given graph similarly as with contrastive learning schemes, but instead of maximizing the similarities across the perturbed graphs as done with contrastive learning, we aim to learn their discrepancies. To this end, we first introduce a discriminator which learns to discriminate real graphs from the perturbed ones (See Discriminator in Figure 1, (c-1)). This allows the model to learn small differences that may largely impact the global property of the graph, as illustrated by the example in Figure 1 (d) and (e). However, simply knowing that two graphs are different is insufficient, and we would want to learn the exact amount of discrepancy between them. Notably, with our perturbation scheme, the exact discrepancy can be trivially obtained since the graph edit distance [26] is automatically derived as the number of edges we perturb for a given graph. Then, we enforce the embedding-level difference between original and perturbed graphs to be proportional to the graph edit distance (See dotted circles in Figure 1 (c-1)). Finally, to discriminate the original and perturbed graphs from the completely different graphs, we enforce the relative distance of the latter to be larger than the former, by a large margin (See blue arrows in Figure 1, (c-1)).

This will enable our D-SLA to capture both small and large topological differences (See Figure 1 (c-2)). Thus, our framework can enjoy the best of both worlds for graph self-supervised learning: predictive and contrastive learning. The experimental results show that our model significantly outperforms baselines, especially on the datasets where graphs with different properties have highly correlated structures, demonstrating the effectiveness of ours. Our main contributions are as follows:

- We propose a novel graph self-supervised learning framework with a completely opposite objective from contrastive learning, which aims to learn to differentiate a graph and its perturbed ones using a discriminator, as even slight perturbations could lead to completely different properties for graphs.
- Utilizing the graph edit distance that is obtained for perturbed graphs at no cost, we propose a novel objective to preserve the exact amount of discrepancy between graphs in the representation space.
- We validate our D-SLA by pre-training and fine-tuning it on various benchmarks of chemical, biology, and social domains, on which it significantly outperforms baselines.

2 Related Work

We now briefly review the existing works on graph neural networks (GNNs), and self-supervised learning methods for GNNs including predictive and contrastive learning.

2.1 Graph Neural Networks

Most existing graph neural networks (GNNs) could be formulated under the message passing scheme [8], which represents each node by firstly aggregating the features from its neighbors, and then combining the aggregated message with its own node representation. Different variants of update and aggregation functions have been studied. To mention a few, Graph Convolutional Network (GCN) [17] generalizes the convolution operation in a spectral domain of graphs, approximated by the mean aggregation. GraphSAGE [9] concatenates the representations of neighbors with its own representation when updating the node representation. Graph Attention Network (GAT) [35] considers the relative importance among neighboring nodes for neighborhood aggregation, which helps identify relevant neighbors for the given task. Graph Isomorphism Network (GIN) [43] uses the sum aggregation, allowing the model to distinguish two different graphs as powerfully as the Weisfeiler-Lehman (WL) test [19]. While GNNs have achieved impressive results on various graph-related tasks, the trained representations from one dataset are usually not transferable to different downstream tasks. Moreover, labels of such task-oriented data are often scarce, especially in scientific domains (e.g., chemistry and biology) [12]. Thus, in this work, we aim at obtaining transferable representations of graph-structured data with self-supervised learning, without using any labels. Self-supervised learning for GNNs can be broadly classified into two categories: predictive learning and contrastive learning, which we will briefly introduce in the following paragraphs.

2.2 Predictive Learning for Graph Self-supervised Learning

Predictive learning aims to learn contextual relationships by predicting subgraphical features, for example, nodes, edges, and subgraphs. Specifically, Hu et al. [12] propose to predict the attributes of masked nodes. Also, Hwang et al. [14] and Kim and Oh [16] propose to predict the presence of an edge or a path with the link prediction scheme. Furthermore, Hu et al. [13] and Rong et al. [25] propose to predict the generative sequence, contextual properties, and motifs of the given graphs. However, the predictive learning methods are limited in that they may not well capture the global structures and/or semantics of graphs.

2.3 Contrastive Learning for Graph Self-supervised Learning

The limitations of predictive learning gave rise to contrastive learning methods, which aim to capture global graph-level information. Early contrastive learning methods for GNNs aim to learn the similarity between the entire graph and its substructure, to obtain the representations for them without using any perturbations [36, 30]. Despite their successes, to further expand the embedding space of a model, contrastive learning methods have been actively studied with perturbation methods such as attribute masking, edge perturbation, and subgraph sampling [47, 52, 37, 48]. Recently proposed adversarial methods further generate positive examples either by adaptively removing the edges [32] or by adjusting the attributes [45]. Further, Xu et al. [44] aim to cluster graph representations via contrastive learning, and Thakoor et al. [33] propose a memory-efficient method for large-scale graph representation learning. However, while contrastive learning allows learning the global semantics with perturbed graphs, there exists a potential risk where the model can consider two dissimilar graphs as the same (See Figure 1 (b), (d), and (e)), due to the discrete nature of graphs. Contrarily, our novel pretext task of predicting the correct graph from multiple choices, which includes both slightly perturbed graphs and strongly perturbed graphs, allows the model to learn the correct relationships among local elements, but also learn the global graph-level differences.

3 Method

In this section, we introduce our novel graph self-supervised learning framework, **D**iscrepancy-based **S**elf-supervised **L**e**A**rning (*D-SLA*), which is illustrated in Figure 2. Specifically, we first recount the notion of graph neural networks and existing graph self-supervised learning in section 3.1. Then, we introduce each component of our D-SLA in Section 3.2, 3.3, and 3.4, respectively. After that, we finally describe the overall framework that combines all three components in Section 3.5.

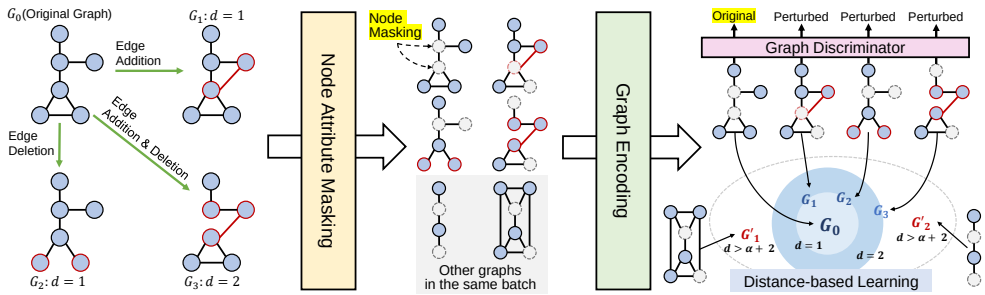


Figure 2: **Illustration of our overall framework (D-SLA).** We perturb the original graph by adding or deleting edges, where the discrepancy between original and perturbed graphs is defined by graph edit distance [26]. Then, we mask the node attributes to make it more difficult to distinguish original and perturbed graphs. After that, we learn GNNs to distinguish perturbed graphs from original ones, but also accurately discriminate the original, perturbed, and other graphs on the embedding space with their distances.

3.1 Preliminaries

Graph Neural Networks A graph $\mathcal{G}$ can be defined by four components: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}_\mathcal{V}, \mathcal{X}_\mathcal{E})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, $\mathcal{X}_\mathcal{V} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the matrix of node features, and $\mathcal{X}_\mathcal{E} \in \mathbb{R}^{|\mathcal{E}| \times d'}$ is the matrix of edge features, where d and d' are the dimensionalities of node and edge attributes, respectively. Let h_v be a representation of a node v, and $h_\mathcal{G}$ be a representation of a graph $\mathcal{G}$. Then, the goal of graph neural networks (GNNs) is to represent nodes of the given graph by leveraging its topological structure. To be specific, GNNs learn node representations (i.e., h_v) by iteratively aggregating messages from their neighbors in a layer-wise manner, which is usually referred to as message-passing [8] and formally defined as follows:

$$h_v^{(l+1)} = f_U^{(l)} \left(h_v^{(l)}, f_A^{(l)} \left(\left\{ h_u^{(l)} : \forall u \in \mathcal{N}(v) \right\} \right) \right), \tag{1}$$

where $f_U^{(l)}$ denotes an update function that updates a representation of the given node along with its neighbors' representations, $f_A^{(l)}$ denotes an aggregate function that aggregates messages from the node's neighbors, $\mathcal{N}(v)$ denotes a set of neighbors of the node v, and l denotes a l-th layer of GNNs.

To further obtain the representation for an entire graph, we usually summarize all node representations from Equation 1 into a single embedding vector with a permutation-invariant function f_R as follows:

$$h_\mathcal{G} = f_R \left(\left\{ h_v^{(L)} : \forall v \in \mathcal{V} \right\} \right). \tag{2}$$

While a natural choice of f_R is to use simple operations, such as mean or sum, various graph representation learning methods have been recently studied to accurately capture the entire graph information, including node clustering based [46, 2], and node drop based methods [7, 18].

Contrastive Learning for Self-supervised Learning of GNNs We now describe why contrastive learning fails to distinguish two topologically similar graphs yet having completely different properties. In particular, contrastive learning aims to increase the similarity between positive pairs of graphs while increasing the dissimilarity between negative pairs [47, 52, 48]. Thereat, to define positive pairs, they perturb the original graph by manipulating edges, masking attributes, and sampling subgraphs, and then consider the perturbed graphs as the same as the original graph. On the other hand, they consider other graphs in the same batch as dissimilar. Formally, for the perturbed graphs $\mathcal{G}_i$ and $\mathcal{G}_j$ from the original graph $\mathcal{G}_0$, the objective of contrastive learning is defined as follows:

$$\mathcal{L}_{CL} = -\log \frac{f_{\text{sim}}(h_{\mathcal{G}_i}, h_{\mathcal{G}_j})}{\sum_{\mathcal{G}',\, \mathcal{G}' \neq \mathcal{G}_0} f_{\text{sim}}(h_{\mathcal{G}_i}, h_{\mathcal{G}'})}, \tag{3}$$

where $\mathcal{G}'$ is the other graph in the same batch with the graph $\mathcal{G}_0$, which is also referred to as a *negative graph*. Therefore, $\mathcal{G}_i$ and $\mathcal{G}_j$ are a positive pair, whereas, $\mathcal{G}_i$ and $\mathcal{G}'$ are a negative pair. f_{sim} denotes a similarity function between two graphs, for example, L_2 distance or cosine similarity. By minimizing the objective in Equation 3, existing contrastive learning methods closely embed two perturbed graphs $\mathcal{G}_i$ and $\mathcal{G}_j$: $\mathcal{G}_i \sim \mathcal{G}_j$. However, as the perturbations may not preserve the properties of the given graph due to the discrete nature of graph-structured data, they indeed should not be similar: $\mathcal{G}_i \not\sim \mathcal{G}_j$ (See Figure 1 (d) and (e)). To this end, we propose a new objective that can differentiate between the original graph and its perturbations, which we describe in the next subsection.

3.2 Discrepancy Learning with Graph Discrimination

We now introduce our novel graph self-supervised learning objective to preserve the discrepancy between two graphs in the learned representation space, for which we then describe the perturbation scheme to generate augmented graphs from the original graph.

Discriminating Original Graphs from Perturbed Graphs Contrarily to contrastive learning that trains perturbed graphs to be similar, our goal is to distinguish the original graph from perturbed graphs, by predicting the original graph among original and perturbed graphs, which we term as the original graph discrimination (See Graph Discriminator in Figure 2). Let $\mathcal{G}_0$ be an original graph, and $\mathcal{G}_i$ be a perturbed graph with $i \geq 1$. Then objective of our discrimination scheme is as follows:

$$\mathcal{L}_{GD} = -\log\left(\frac{e^{S_0}}{e^{S_0} + \sum_{i\geq 1} e^{S_i}}\right) \text{ with } S = f_S(h_{\mathcal{G}}), \tag{4}$$

where S_k is a score of the graph $\mathcal{G}_k$, obtained from a learnable score network f_S of the discriminator, i.e., $f_S : h_{\mathcal{G}} \mapsto S \in \mathbb{R}$. By training to discriminate the original graph from the perturbed graphs, the model is enforced to embed the perturbed graphs apart from the original graph (Figure 3 (b)). Therefore, the model learned by our discrimination scheme will be capable of distinguishing even the slight differences between the original graph and its perturbations, as well as be capable of capturing the correct distribution of the graphs, since perturbed graphs could be semantically incorrect.

Perturbation The remaining question is how to perturb the graph to generate negative examples (i.e., $\mathcal{G}_i$). We aim at adding or deleting the *subset* of edges in the given graph, and thus such perturbed graphs are slightly different from the original graph but also could be semantically incorrect. Again, in this way, since our graph discrimination objective in Equation 4 discriminates the original graph from slightly perturbed graphs, the graph-level representations from our pre-trained GNNs can capture subtle difference that may have a large impact on the properties of graphs, for downstream tasks.

In particular, our perturbation scheme consists of the following two steps: 1) removing and adding a small number of edges, and 2) masking node attributes (Figure 2). To be specific, given an original graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}_\mathcal{V}, \mathcal{X}_\mathcal{E})$, we aim to perturb it n times, to obtain $\{\mathcal{G}_1, ..., \mathcal{G}_n\}$. To do so, we first manipulate the edge set $\mathcal{E}$ by removing existing edges as well as adding new edges on it and then adjust its corresponding edge matrix $\mathcal{X}_\mathcal{E}$. After that, we further randomly mask the node attributes on $\mathcal{X}_\mathcal{V}$ for both original and perturbed graphs, to make it more difficult to distinguish between them. Formally, the original and perturbed graphs from our node/edge perturbations are obtained as follows:

$$\begin{aligned} \mathcal{G}_0 &= \left(\mathcal{V}, \mathcal{E}, \tilde{\mathcal{X}}^0_\mathcal{V}, \mathcal{X}_\mathcal{E}\right),\ \tilde{\mathcal{X}}^0_\mathcal{V} \sim \mathtt{M}(\mathcal{G}), \\ \mathcal{G}_i &= \left(\mathcal{V}, \mathcal{E}^i, \tilde{\mathcal{X}}^i_\mathcal{V}, \mathcal{X}^i_\mathcal{E}\right),\ \tilde{\mathcal{X}}^i_\mathcal{V} \sim \mathtt{M}(\mathcal{G}),\ (\mathcal{E}^i, \mathcal{X}^i_\mathcal{E}) \sim \mathtt{P}(\mathcal{G}), \end{aligned} \tag{5}$$

where $\mathtt{M}$ and $\mathtt{P}$ are the node masking and edge perturbation functions, respectively. The scores of original and perturbed graphs are then computed with the score function f_S: $[S_0, S_1, ..., S_n] = [f_S(\mathcal{G}_0), f_S(\mathcal{G}_1), ..., f_S(\mathcal{G}_n)]$, which are then used for our previous learning objective in Equation 4.

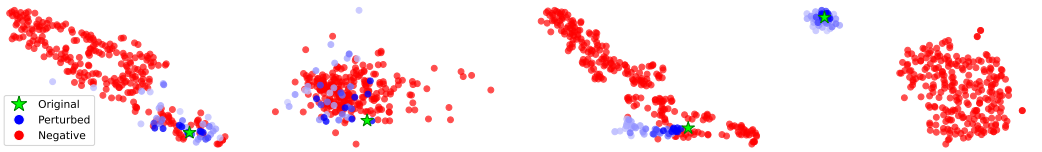
3.3 Learning Discrepancy with Edit Distance

While the learning to discriminate original and perturbed graphs described in Section 3.2 allows the model to learn the discrepancy between original and perturbed graphs by embedding them onto different points, it cannot learn *how dissimilar* the perturbed graph is from the original one as all perturbed graphs are considered as equal (Figure 3 (b)). Thus, to further learn the accurate discrepancy among different graphs, here we introduce a method for efficiently computing the graph discrepancy between original and perturbed graphs leveraging the graph edit distance [26]. Then we describe how to preserve the calculated discrepancy in the learned graph representation space.

Graph Edit Distance To preserve the exact amount of discrepancies between original and perturbed graphs in the learned representation space, we first need to measure the graph distance. Graph edit distance, which is widely used to measure the dissimilarity between two graphs [26], is defined by the number of insertion, deletion, and substitution operations for nodes and edges, to transform one graph into the other graph. Although computing

Table 1: Costs for calculating distance, with m edges and h iterations for WL-kernel [28].

Methods	Costs
Graph edit distance [26]	NP-hard [50]
Weisfeiler-Lehman kernel [28]	$\mathcal{O}(hm)$
Graph edit distance **with our perturbation**	$\mathcal{O}(1)$



(a) Randomly Init. (b) $\mathcal{L}_{GD}$ (c) $\mathcal{L}_{GD} + \mathcal{L}_{edit}$ (d) $\mathcal{L}_{GD} + \mathcal{L}_{edit} + \mathcal{L}_{margin}$

Figure 3: Visualizations of the graph embeddings to see the effect of each loss for our D-SLA. We represent the level of perturbations via the transparency of color, i.e., the stronger the perturbation, the lighter the color.

the graph edit distance is the NP-hard problem [50], for our case, it is trivially obtained without any cost. This is because, for each perturbed graph, we know the exact number of edge addition and deletion steps as we have generated it using the same process. In other words, the number of added and deleted edges is simply the edit distance between original and perturbed graphs. This brings us significant computational advantages, as we generally require a significant amount of costs for calculating or estimating the distances between two graphs, as shown in Table 1. Note that although several existing works [20, 3] leverage graph edit distance, they are completely different from our work since they are neither self-supervised learning methods nor leverage the graph perturbation.

Distance-based Discrepancy Learning Based upon the graph edit distance, we design the regularization term to learn the exact amount of differences between original and perturbed graphs over the embedding space. To be specific, we regularize the model to learn that the embedding-level difference between original and perturbed graphs is proportional to their actual graph edit distance (i.e., if the edit distance between two graphs is large, then they are far away in the embedding space, whereas, if it is small, then they are close to each other) as shown in Distance-based Learning of Figure 2.

We first let an original graph $\mathcal{G}_0$ be an anchor graph, then a graph edit distance and an embedding-level distance between the anchor graph $\mathcal{G}_0$ and the perturbed graph $\mathcal{G}_i$ are defined as e_i and d_i, respectively. With such notations, to achieve our objective of learning an exact amount of discrepancy across original and perturbed graphs, we formalize our edit distance-based regularization loss as follows:

$$\mathcal{L}_{edit} = \sum\nolimits_{i,j} \left(\frac{d_i}{e_i} - \frac{d_j}{e_j} \right)^2 \text{ with } d_i = f_{\text{diff}}(h_{\mathcal{G}_0}, h_{\mathcal{G}_i}), \tag{6}$$

where f_{diff} measures the embedding-level differences between graphs with L_2-norm. We find that Equation 6 can capture the exact amount of differences as shown in Figure 3 (c). Note that Equation 6 alone does not work as shown in Table 5, since the trivial solution of Equation 6 is to set all the embedding-level distance between original and perturbed graphs as zero (i.e., $d_i = 0 \; \forall i$). However, learning the accurate amount of discrepancy without the trivial solution is feasible by jointly training with the objective of original graph discrimination in Equation 4 where the model should differently embed $\mathcal{G}_0$ and $\mathcal{G}_i$ to discriminate them. Furthermore, as we can generate different levels of perturbations according to the graph edit distance, our graph embedding space can interpolate between two different graphs, for example, weakly and strongly perturbed graphs in terms of their actual distances, unlike previous predictive and contrastive learning methods.

3.4 Relative Discrepancy Learning with Other Graphs

Thus far, we aimed toward distinguishing the perturbed graphs from the original graph in Section 3.2, with their actual edit distance-based discrepancy learning in Section 3.3. However, such schemes alone will not allow the model to learn the discrepancy between two completely different graphs (Figure 3 (c)). While one might consider learning the discrepancy between original and its negative graphs (other graphs in the same batch) with their graph edit distance, it is impractical as computing the edit distance between completely different graphs is NP-hard [50]. Thus, we instead propose to learn a relative distance, exploiting the assumption that the distance between original and negative graphs in the same batch is larger than the distance between original and perturbed graphs with some amount of *margin* (See Figure 2, Distance-based Learning). The usage of margin is highly beneficial for our discrepancy-based learning framework, since, if the negative graphs are far apart than the amount of margin plus the distance between original and perturbed graphs, the model does not attract the perturbed graphs to the original graph, therefore not losing the discrepancy learned in Equation 6.

Formally, we realize the relative discrepancy learning with the triplet margin loss as follows:

$$\mathcal{L}_{margin} = \sum\nolimits_{i,j} \max(0, \; \alpha + d_i - d'_j), \tag{7}$$

where d_i is the distance between original and its perturbed graphs: $d_i = f_{\text{diff}}(h_{\mathcal{G}_0}, h_{\mathcal{G}_i})$, d'_j is the distance between original and its negative graphs: $d'_j = f_{\text{diff}}(h_{\mathcal{G}_0}, h_{\mathcal{G}'_j})$ with $\mathcal{G}'$ as in-batch negative graphs, and $\alpha > 0$ as a margin hyperparameter. In this way, we can allow the model to learn that, for the original graph, negative graphs are more dissimilar than the perturbed graphs, while perturbed graphs are also marginally dissimilar to it with the amount of edit distances as shown in Figure 3 (d).

3.5 Overall Framework

To sum up, our D-SLA framework aims to learn a graph representation space which preserves the discrepancy among graphs. Our graph discriminator described in Section 3.2 helps learn a space that can discriminate even the slightly perturbed graph from its original graph (Figure 3 (b)). Then, the discrepancy learning with graph edit distance described in Section 3.3 (Figure 3 (c)) enforces the embedding space to preserve the exact discrepancy among graphs. Finally, the margin-based triplet constraints between the perturbed and completely different graphs in Section 3.4 allow the learned embedding space to capture the relative distance between graphs (Figure 3 (d)). Our overall learning objective, dubbed as Discrepancy-based Self-supervised LeArning (D-SLA), is given as follows:

$$\mathcal{L} = \mathcal{L}_{GD} + \lambda_1 \mathcal{L}_{edit} + \lambda_2 \mathcal{L}_{margin}, \tag{8}$$

where hyperparameters λ_1 and λ_2 are scaling weights to each loss. Note that, unlike predictive learning [12] which aims to capture local semantics of graphs by masking and predicting local components, such as nodes and edges, our D-SLA learns the graph-level representations where the subtle differences on corner regions of different graphs could be captured by distinguishing original and perturbed graphs. Moreover, our D-SLA can discriminate differently perturbed graphs having different properties, unlike contrastive learning [47, 48] that considers them as similar.

4 Experiments

In this section, we first experimentally validate the proposed Discrepancy-based Self-supervised LeArning (D-SLA) on graph classification tasks to verify its effectiveness in obtaining accurate graph-level representations. After that, we further evaluate our D-SLA on link prediction tasks for which capturing the local semantics of graphs is important.

4.1 Graph Classification

Accurately capturing the global semantics of given graphs is crucial for graph classification tasks, on which we validate the performance of our D-SLA against existing baselines.

Experimental Setup We evaluate our D-SLA on two different domains: the molecular property prediction task from chemical domain [38, 11] and the protein function prediction task from biological domain [12]. For the chemical domain, we follow the experimental setup from You et al. [48], whose goal is to predict the molecules' biochemical activities. For the self-supervised learning, we use 2M molecules from the ZINC15 dataset [29]. Then, after the self-supervised learning, we perform fine-tuning on datasets from MoleculeNet [38] to evaluate the down-stream performances of models. For the biological domain, we follow the setup from You et al. [48], for which the goal is to predict the proteins' biological functions. For pre-training and fine-tuning, we use the dataset of PPI networks [53]. We use the ROC-AUC value as an evaluation metric, and report the average performance over five different runs following Xu et al. [44]. See Appendix A.2 for dataset details.

Models We compare our D-SLA against predictive learning baselines: EdgePred [9], AttrMasking [12] and ContextPred [12], and contrastive learning baselines: Infomax [36], GraphCL [47], JOAO [48], GraphLoG [44], and BGRL [33]. Detailed explanations of models including baselines and ours are provided in Appendix A.1.

Implementation Details We follow the conventional experimental setup of graph self-supervised learning from Hu et al. [12], where we use the GIN [43] as the base network. For pre-training of our model, we perturb the original graph three times. To be specific, to obtain the perturbed graphs from the original graph, we first randomly select a subgraph of it, and then add or remove edges in the range of $\{20\%, 40\%, 60\%\}$ over the sampled subgraph. For fine-tuning, we follow the hyperparameters from You et al. [48]. We provide additional details in Appendix A.2.

Table 2: Fine-tuning results on graph classification tasks of chemical and biological domains. Best performances are highlighted in bold. The reported results are taken from You et al. [48] and Xu et al. [44], except for the BGRL results and the PPI performance of GraphLoG as they are not available.

	SSL methods	BBBP	ClinTox	MUV	HIV	BACE	SIDER	Tox21	ToxCast	PPI	Avg.
	No Pretrain	65.8 ± 4.5	58.0 ± 4.4	71.8 ± 2.5	75.3 ± 1.9	70.1 ± 5.4	57.3 ± 1.6	74.0 ± 0.8	63.4 ± 0.6	64.8 ± 1.0	66.72
Predictive	Edgepred	67.3 ± 2.4	64.1 ± 3.7	74.1 ± 2.1	76.3 ± 1.0	79.9 ± 0.9	60.4 ± 0.7	76.0 ± 0.6	64.1 ± 0.6	65.7 ± 1.3	69.77
Predictive	AttrMaskig	64.3 ± 2.8	71.8 ± 4.1	74.7 ± 1.4	77.2 ± 1.1	79.3 ± 1.6	61.0 ± 0.7	76.7 ± 0.4	64.2 ± 0.5	65.2 ± 1.6	70.49
Predictive	ContextPred	68.0 ± 2.0	65.9 ± 3.8	75.8 ± 1.7	77.3 ± 1.0	79.6 ± 1.2	60.9 ± 0.6	75.7 ± 0.7	63.9 ± 0.6	64.4 ± 1.3	70.17
Contrastive	Infomax	68.8 ± 0.8	69.9 ± 3.0	75.3 ± 2.5	76.0 ± 0.7	75.9 ± 1.6	58.4 ± 0.8	75.3 ± 0.5	62.7 ± 0.4	64.1 ± 1.5	69.60
Contrastive	GraphCL	69.68 ± 0.67	75.99 ± 2.65	69.80 ± 2.66	78.47 ± 1.22	75.38 ± 1.44	60.53 ± 0.88	73.87 ± 0.66	62.40 ± 0.57	67.88 ± 0.85	70.44
Contrastive	JOAO	70.22 ± 0.98	**81.32** ± 2.49	71.66 ± 1.43	76.73 ± 1.23	77.34 ± 0.48	59.97 ± 0.79	74.98 ± 0.29	62.94 ± 0.48	64.43 ± 1.38	71.07
Contrastive	JOAOv2	71.39 ± 0.92	80.97 ± 1.64	73.67 ± 1.00	77.51 ± 1.17	75.49 ± 1.27	60.49 ± 0.74	74.27 ± 0.62	63.16 ± 0.45	63.94 ± 1.59	71.21
Contrastive	GraphLoG	72.5 ± 0.8	76.7 ± 3.3	76.0 ± 1.1	77.8 ± 0.8	83.5 ± 1.2	**61.2** ± 1.1	75.7 ± 0.5	63.5 ± 0.7	66.92 ± 1.58	72.65
Contrastive	BGRL	66.73 ± 1.70	64.74 ± 6.46	69.36 ± 2.66	75.52 ± 1.85	71.27 ± 5.48	60.41 ± 1.43	74.83 ± 0.74	63.20 ± 0.76	64.69 ± 1.66	67.86
	D-SLA (Ours)	**72.60** ± 0.79	80.17 ± 1.50	**76.64** ± 0.91	**78.59** ± 0.44	**83.81** ± 1.01	60.22 ± 1.13	**76.81** ± 0.52	**64.24** ± 0.50	**71.56** ± 0.46	**73.85**

Dataset	Act. Sim.	Inact. Sim.
BACE	**0.6743**	**0.5403**
HIV	0.4186	0.4536
MUV	0.1946	0.4181
Tox21	0.3047	0.3462
ToxCast	0.2193	0.2962
SIDER	0.2880	0.2316
ClinTox	0.2725	0.2278
BBBP	0.3961	0.2031

Table 3: Tanimoto similarity on chemical benchmark datasets.

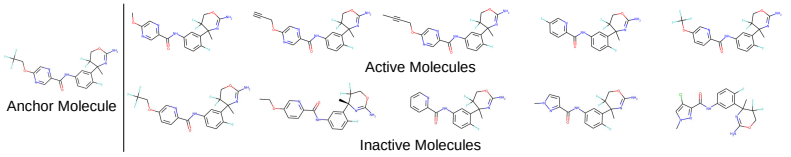


Figure 4: Visualization of molecules in BACE, where the active and inactive molecules in the top and bottom sides are the most similar ones to the molecule in the left, according to the Tanimoto similarity.

Results Table 2 shows that our D-SLA achieves the best average performance against existing predictive and contrastive learning baselines on tasks from both chemical and biological domains, demonstrating the effectiveness of our discrepancy-based framework. To better see what aspects of D-SLA contribute to the performance improvements, we perform in-depth analyses of each dataset, and how our model can effectively handle the task on it, in the next paragraph.

Analysis To analyze whether the structural similarity of molecules is correlated to their biochemical activities, we measure the inherent discrepancy of graphs with the Tanimoto similarity over Morgan fingerprints [24]. To be specific, we first iteratively sample an anchor molecule among active molecules in the dataset, and then measure the average Tanimoto similarities of the five most similar active/inactive molecules. In other words, the high similarity values of Inact. Sim. in Table 3 suggests that the molecules have highly overlapped structures regardless of their biochemical activities. For example, as shown in Table 3, the molecules in the BACE dataset are highly correlated, although their activities are different. Also, we further observe that, as shown in Figure 4 that visualizes the most similar active/inactive molecules with respect to the certain anchor molecule in the BACE dataset, the structures between active and inactive molecules are highly similar.

From the above observations, we suggest that, due to the discrete nature of graphs, two slightly different graphs can have completely different properties, which may be the reason for the performance degeneration of contrastive learning methods in Table 2 – which consider perturbed graphs as similar while they might not be the semantically same – on such particular datasets (e.g., BACE, MUV, Tox21, ToxCast) having the high Inact. Sim. scores in Table 3. However, our D-SLA largely outperforms contrastive learning baselines on them[2], because ours not only discriminates an original graph from its perturbations but also can learn their exact discrepancy via the graph edit distance. We note that our method shows the competitive performance on ClinTox and SIDER in Table 2, since they have the lowest structural similarities across different biochemical properties, for which contrastive learning could be effective. We provide more analysis in Appendix B.1.

4.2 Link Prediction

Accurately capturing the local semantics of a graph is an important requisite for solving node/edge-level tasks. Thus, we further validate our D-SLA on the link prediction task.

Experimental Setup We conduct link prediction experiments on social network datasets – COLLAB, IMDB-B, and IMDB-M – from the TU benchmarks [21]. We separate the dataset into four parts: pre-training, training, validation, and test sets in the ratio of 5:1:1:3. We use the average

[2]GraphLoG also adopts the predictive learning scheme by matching the locally masked nodes/subgraphs to their original substructures, thus it shows decent performances on such high similarity datasets.

	SSL Method	COLLAB	IMDB-B	IMDB-M	Avg.
	No Pretrain	80.01 ± 1.14	68.72 ± 2.58	64.93 ± 1.92	71.22
Pred	AttrMasking	81.43 ± 0.80	70.62 ± 3.68	63.37 ± 2.15	71.81
	ContextPred	83.96 ± 0.75	70.47 ± 2.24	66.09 ± 2.74	73.51
Contra	Infomax	80.83 ± 0.62	67.25 ± 1.87	64.98 ± 2.47	71.02
	GraphCL	76.04 ± 1.04	63.71 ± 2.98	62.40 ± 3.04	67.38
	JOAO	76.57 ± 1.54	65.37 ± 3.23	62.76 ± 1.52	68.23
	GraphLoG	82.95 ± 0.98	69.71 ± 3.18	64.88 ± 1.87	72.51
	BGRL	76.79 ± 1.13	67.97 ± 4.14	63.71 ± 2.09	69.49
	D-SLA (Ours)	**86.21** ± 0.38	**78.54** ± 2.79	**69.45** ± 2.29	**78.07**

Table 4: Link prediction results on the social network datasets. The reported results are the average precision, and the bold number denotes the best performance on each dataset.

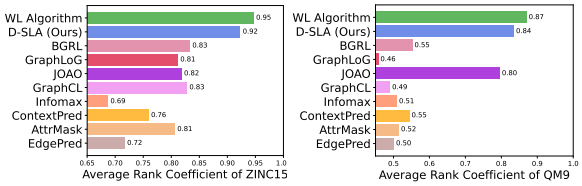


Figure 5: Rank correlation coefficient of 1,000 graphs from ZINC15 and QM9 datasets, measuring the coefficient between the actual similarity ranks and the calculated similarity ranks, between original and perturbed graphs.

precision as an evaluation metric, and report the results over five different runs. We provide more details in Appendix A.3.

Implementation Details For GNNs, we use the GCN [17] following You et al. [48]. For perturbation, we add or delete only a tiny amount of edges (e.g., 1 or 2 edges) while increasing the magnitude of perturbation to obtain three perturbed graphs. Note that we do not use the margin triplet loss in Section 3.4, since for local prediction tasks, the graph-level discrepancy learning between completely different graphs is not much helpful in capturing local semantics. For fine-tuning, we train the GNNs to predict whether there is an edge between nodes, and evaluate the GNNs for predicting the existence of 10 edges. For more implementation details, please see Appendix A.3.

Results As shown in Table 4, predictive learning baselines outperform other baselines, since it learns to predict local node/edge attributes. However, our D-SLA largely outperforms all baselines, while it aims to discriminate graph-level representations though. We suggest that this is because ours can capture subtle differences of graphs by leveraging the graph edit distance, demonstrating that accurate discrepancy learning is obviously useful for the local link prediction task.

4.3 Analysis

In this section, we further analyze the efficacy of our D-SLA. We provide the additional experimental details in Appendix A.4.

Rank Correlation Coefficient To see whether learned representations capture the exact amount of discrepancy, we compare the ranks of the calculated vs actual similarities between original and perturbed graphs with Spearman's rank correlation coefficient. Note that the results of the WL algorithm [19] are merely a performance indicator of discriminative power since it cannot obtain representations that generalize to downstream tasks. As shown in Figure 5, unlike baselines that mostly fail to discriminate different graphs, our D-SLA has the discriminative power that is on par with the powerful WL algorithm, while ours can generalize to downstream tasks as shown in Table 2.

Ablation Study To see how much each component contributes to the performance gain, we conduct an ablation study. As shown in Table 5, we observe that our graph discrimination task ($\mathcal{L}_{GD}$) significantly improves the performance on down-stream tasks against no-pretraining. However, leveraging the exact edit distance alone ($\mathcal{L}_{edit}$) does not learn meaningful representations, since the model trivially sets all the distances between original and its perturbations as zero, as discussed in Section 3.3. Also, we further observe that each component of distance-based learning in Section 3.3 and 3.4 helps to improve the performance, verifying that accurate discrepancy learning with edit and relative distances is important for modeling graphs. We further discuss the ablation study in Appendix B.2.

Table 5: Ablation study for our D-SLA on graph classification and link prediction tasks.

$\mathcal{L}_{GD}$	$\mathcal{L}_{edit}$	$\mathcal{L}_{margin}$	ClinTox	BACE	COLLAB
			58.00	70.10	71.21
✓			70.83	81.58	74.23
	✓		57.46	69.99	72.61
✓	✓		77.48	83.53	76.19
✓	✓	✓	80.17	83.81	N/A

Embedding Visualization We visualize the embedding space from various pre-training methods: predictive, contrastive, and ours, in Figure 1 (a,b,c-2). We observe that predictive learning cannot capture the global graph-level similarity well, as it aims to predict subgraphical semantics of graphs during the pretext task. While contrastive learning can closely embed the highly similar graphs to the original graph, it cannot accurately capture the exact amount of discrepancies among perturbed graphs. Contrarily, our D-SLA can accurately distinguish between the original, perturbed, and negative graphs, while accurately capturing the exact amount of discrepancy for the perturbed graphs.

5 Conclusion

In this work, we focused on the limitations of existing self-supervised learning for GNNs: predictive learning does not capture the graph-level similarities; contrastive learning might treat two semantically different graphs from perturbations as similar. To overcome such limitations, we proposed a novel framework (D-SLA) that can learn the graph-level differences among different graphs while also can learn the slight edge-wise differences, by discriminating the original from perturbed graphs. Further, the model is trained to differentiate the target graph from its perturbations and other graphs, while preserving the accurate graph edit distance, allowing the model to discriminate between not only two structurally different graphs but also similar graphs with slight differences. We validated our D-SLA on 12 benchmark datasets, achieving the best average performance. Further analysis shows that it learns a discriminative space of graphs, reflecting the graph edit distances between them.

6 Limitation and Potential Societal Impacts

In this section, we discuss the limitation and potential societal impacts of our work.

Limitation In this work, we propose a graph self-supervised learning framework, which aims to learn the discrepancy between the original and perturbed graphs, by discriminating the original graph among the original and its perturbations, but also by learning the exact amount of discrepancy across them with the graph edit distance. Note that, while our perturbation scheme can easily make the slight structural differences, it cannot be aware of the semantic differences across differently perturbed graphs, since the graph semantics depend on the target domain. In other words, nodes and edges could differently contribute to the graph semantics depending on the target graph domain (e.g., molecular graphs or social networks), which are hard to pre-define in one framework. As manually annotating the semantic differences across different graphs is impossible to do in an end-to-end fashion, one may need to devise a clever way to be aware of the semantics of graphs for further reflecting the exact semantic difference in the embedding space, which we leave as future work.

Potential Societal Impacts Discovering de novo drugs is significantly important to our society since they can be used for curing a disease or enhancing agricultural production, that is directly related to our life. However, it is extremely costly and time-consuming to design drug candidates and validate them, requiring immense lab experiments and labor. In this work, we verify our proposed method in the graph classification task in Table 2 of the main paper, and ours outperforms the other baselines for classifying toxicity on the Tox21 and ToxCast datasets. We strongly believe that such aspects of our method can positively contribute to our society in developing valuable drugs. However, when validating the toxicity of the most acceptable drug candidates, someone might badly use our method to reduce the money and time (i.e., does not check the toxicity of drugs for reducing costs, however, which may be used for clinical trials to humans), which can negatively affect our society. We sincerely hope that our method would not be utilized for such a bad purpose.

7 Acknowledgements and Disclosure of Funding

We thank anonymous reviewers for their constructive feedback. This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program (KAIST), and No.2021-0-02068, Artificial Intelligence Innovation Hub), the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921), and Samsung Electronics (IO201214-08145-01).

References

[1] Jinheon Baek, Dong Bok Lee, and Sung Ju Hwang. Learning to extrapolate knowledge: Transductive few-shot out-of-graph link prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[2] Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate learning of graph representations with graph multiset pooling. In *International Conference on Learning Representations (ICLR)*, 2021.

[3] Yunsheng Bai, Hao Ding, Ken Gu, Yizhou Sun, and Wei Wang. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[5] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference (WWW)*, 2019.

[6] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[7] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

[8] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[9] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[11] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[12] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[13] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. GPT-GNN: generative pre-training of graph neural networks. In *Conference on Knowledge Discovery and Data Mining (KDD)*, 2020.

[14] Dasol Hwang, Jinyoung Park, Sunyoung Kwon, Kyung-Min Kim, Jung-Woo Ha, and Hyunwoo J. Kim. Self-supervised auxiliary learning with meta-paths for heterogeneous graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[15] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[16] Dongkwan Kim and Alice H. Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations (ICLR)*, 2021.

[17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[18] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

[19] AA Leman and B Weisfeiler. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9):12–16, 1968.

[20] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International Conference on Machine Learning (ICML)*, 2019.

[21] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint*, arXiv:2007.08663, 2020.

[22] Giulia Muzio, Leslie O'Bray, and Karsten M. Borgwardt. Biological network analysis with deep learning. *Briefings Bioinform.*, 22(2):1515–1530, 2021.

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[24] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *J. Chem. Inf. Model.*, 50 (5):742–754, 2010.

[25] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[26] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (3):353–362, 1983.

[27] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[28] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[29] Teague Sterling and John Irwin. Zinc 15 - ligand discovery for everyone. *Journal of chemical information and modeling*, 55, 10 2015. doi: 10.1021/acs.jcim.5b00559.

[30] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations (ICLR)*, 2020.

[31] 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 *The Web Conference (WWW)*, 2021.

[32] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. *arXiv preprint arXiv:2106.05819*, 2021.

[33] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations (ICLR)*, 2022.

[34] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[35] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[36] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations (ICLR)*, 2019.

[37] Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. Molecular contrastive learning of representations via graph neural networks. *Nature Machine Intelligence*, 4(3): 279–287, 2022.

[38] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[39] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[40] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z. Li. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *ACM Web Conference (WWW)*, 2022.

[41] Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. MARS: markov molecular sampling for multi-objective drug discovery. In *International Conference on Learning Representations (ICLR)*, 2021.

[42] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. Infogcl: Information-aware graph contrastive learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[43] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[44] Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. Self-supervised graph-level representation learning with local and global structure. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[45] Longqi Yang, Liangliang Zhang, and Wenjing Yang. Graph adversarial self-supervised learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[46] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems(NeurIPS)*, 2018.

[47] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[48] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[49] 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 *International Conference on Research and Development in Information Retrieval (SIGIR)*, 2022.

[50] Zhiping Zeng, Anthony K. H. Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. Comparing stars: On approximating graph edit distance. *Proc. VLDB Endow.*, 2(1):25–36, 2009.

[51] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*, 2020.

[52] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *The Web Conference (WWW)*, 2021.

[53] Marinka Zitnik, Rok Sosič, Marcus W. Feldman, and Jure Leskovec. Evolution of resilience in protein interactomes across the tree of life. *Proceedings of the National Academy of Sciences*, 116(10):4426–4433, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1818013116.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discuss the limitation in Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss the potential societal impacts in Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide the code for reproducing our results.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provide the detailed experimental setups in section A of Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All of our main results contain variances or standard deviations of multiple runs.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We provide the type of GPUs in Section A of Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We uses the public code and data, and we provide the citation and URLs of them in Section A of Appendix.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

Organization In Section A, we first introduce the baselines and our model and then describe the experimental details of graph classification and link prediction tasks but also our in-depth analyses. Then, in Section B, we provide the additional experimental results about analyses on datasets, ablation study for our proposed objectives, effects of our hyperparameters (λ_1, α, λ_2, and the perturbation magnitude), ablation study of attribute masking, and the comparison with augmentation-free approaches.

A Experimental Details

In this section, we first introduce the computing resources that we use, the baselines, and our model in Section A.1. After that, we describe the experimental setups of the graph classification and link prediction tasks in Section A.2 and Section A.3 as well as the analysis in Section A.4.

Computing Resources For all experiments, we use PyTorch and PyTorch Geometric libraries [23, 6], for easy usage of GPU resources. We use TITAN XP and GeForce RTX 2080 Ti for training and evaluating all models.

A.1 Baselines and Our Model

1. **EdgePred** is a predictive learning baseline adopted from the link prediction task of Hamilton et al. [9], whose goal is to predict the existence of edges between the given two nodes.
2. **AttrMasking** [12] is a predictive learning baseline that predicts the attributes of masked nodes and edges from the embeddings of nodes.
3. **ContxtPred** [12] is a predictive learning baseline that first samples two different subgraphs from the same centered node, and then trains them to be similar while the subgraphs from the other graphs are trained to be dissimilar.
4. **Infomax** [36] is a contrastive learning baseline, whose goal is to learn the representations for the given graph and the substructure within the same given graph to be similar while learning the representations for the given graph and the substructures from the negative graphs to be dissimilar.
5. **GraphCL** [47] is a contrastive learning baseline, whose goal is to learn the similarity between two perturbed graphs from the same graph contrasting to in-batch negative graphs over the global graph-level representations. In particular, this method uses the following four perturbation methods: attribute masking, edge perturbing, node dropping, and subgraph sampling.
6. **JOAO** [48] is a contrastive learning baseline that, while the learning objective of it is the same as the GraphCL model described above, learns to automatically select the perturbation schemes.
7. **JOAOv2** [48] is a variant of JOAO, which has individual projection heads according to the perturbation schemes. Specifically, a perturbed graph is fed into the typical projection head according to the selected perturbation.
8. **GraphLoG** [44] is a baseline that has two learning objectives: 1) it matches the masked nodes/graphs to their unmasked counterparts; 2) it clusters a group of globally similar graphs with learnable cluster prototypes.
9. **BGRL** [33] is a baseline that maximizes the similarity between two perturbed graphs from the original graph without considering in-batch negative graphs, aiming to represent large-scale graphs with efficiency in memory usage.
10. **D-SLA** is our discrepancy-based graph self-supervised learning framework, which aims to learn the accurate discrepancy between original, perturbed, and negative graphs, by not only discriminating the original graph from its perturbations but also preserving the accurate amount of discrepancy with the graph edit distance between them.

A.2 Graph Classification

Datasets We use the available benchmark datasets[3] for the graph classification task. Specifically, for the chemical domain, we use 2M molecules sampled from the ZINC15 dataset [29] without using any labels on it. The fine-tuning datasets consist of the molecular graphs from MoleculeNet [38], where the classes are given by the biophysical and physiological properties of the molecules. For the biological domain, the datasets are constructed by the sampled ego-networks from the PPI networks [53]. In particular, the pre-training dataset consists of 306K unlabeled protein ego-networks of 50 species, and the fine-tuning dataset consists of 88K protein ego-networks of 8 species with the label given by the functionality of the ego protein. We report the statistics of graph classification datasets in Table 6.

Table 6: Dataset statistics on chemical and biological domains.

Dataset	Tasks	Graphs	Avg. Nodes	Avg. Edges
Chemical Domain				
ZINC15 (Pre-training)	-	2,000,000	26.62	28.86
QM9 (Rank Coeff.)	-	133,149	8.80	9.40
BBBP	1	2,039	24.06	25.95
ClinTox	2	1,478	26.16	27.88
MUV	17	93,087	24.23	26.28
HIV	1	41,127	25.51	27.47
BACE	1	1,513	34.09	36.86
SIDER	27	1,427	33.64	35.36
Tox21	12	7,831	18.57	19.29
ToxCast	617	8,575	18.78	19.26
Biological Domain				
PPI (Pre-training)	-	306,925	39.83	364.82
PPI (Fine-tune)	40	88,000	49.35	445.39

Strategy for Selecting Edges for Perturbations In this paragraph, we describe the detailed edge selection scheme for our graph perturbation. In our experiments of graph classification, we first select the node and then sample the 3-hop subgraph of it. After that, we randomly add and remove edges on the subgraph. The reason behind selecting the target subgraph for perturbation is that we aim to reduce the potential risk of making unreasonable cycles, which are impractical especially on the chemical domain. Therefore, to prevent the model to learn such an incorrect bias in the embedding space, we rather sample the subgraph for perturbing the edges.

Common Implementation Details We follow the conventional design choice of GNNs for evaluating the graph self-supervised learning methods from Hu et al. [12]: Graph Isomorphism Networks (GINs) [43] consisting of 5 layers with 300 dimensions along with mean average pooling for obtaining the entire graph representations. For pre-training of our D-SLA, we sample a subgraph by randomly selecting a center node and then select 3-hop neighbors of it, and then remove the edges on the selected subgraph three times with different magnitudes (20%, 40%, 60%) to make three perturbed graphs, while memorizing the number of deleted and added edges to calculate the graph edit distance. To prevent the situation where the deleted edges are added again, we add edges that are not present in the given original graphs. We mask 80% of nodes in the selected subgraph to confuse the model to distinguish the original graph from its perturbed graphs. Furthermore, we include the strong perturbation, where 80% of edges are perturbed and 80% of nodes are masked among entire nodes and edges in the given graph. λ_1 and λ_2 are set as 0.7 and 0.5, respectively.

Implementation Details on Molecular Property Prediction We follow the conventional molecule representation setting from Hu et al. [12], where the node attributes contain the atom number along with the chirality, and the edge attributes contain the bond type (e.g., Single, Double, Triple or Aromatic) along with the bond direction which is represented if an edge is a double or aromatic bond. When adding an edge during edge perturbation, we sample its type by following the distribution of edge attributes in the pre-training dataset. Specifically, we first sample the bond type following the distribution and then sample also the bond direction depending on the bone type. For pre-training, we use the batch size of 256, the number of epochs of 100, the learning rate in the range of [0.01, 0.001, 0.0001], and the margin α in the range of [3,4,5,6,7] by grid search. For the splitting of fine-tuning datasets, we use the scaffold splitting following the conventional setting from Hu et al. [12] and You et al. [48]. For fine-tuning, we also follow the conventional setting from You et al. [48].

Implementation Details on Protein Function Prediction We use the pre-defined biological graphs from Hu et al. [12], where a node corresponds to a protein without any attributes, and an edge corresponds to a relation type between two proteins such as biological interaction or co-expression. As in molecular property prediction, we add reasonable edges by following the distribution of edge attributes in the pre-training dataset. For pre-training, the number of epochs is 100, the batch size is 128, the learning rate is 0.001, and the margin is 10. For data splitting of the fine-tuning dataset, we use the provided conventional setting from Hu et al. [12]. For fine-tuning, we also follow the conventional setting from Hu et al. [12]. Note that, as the result of GraphLoG [44] on this protein function prediction task is not available in the referred paper, we produce the result by following the experimental setups along with the provided public source code.

[3] http://snap.stanford.edu/gnn-pretrain/data/

A.3 Link Prediction

Datasets The datasets[4] we used for the link prediction task are COLLAB, IMDB-B (IMDB-BINARY), IMDB-M (IMDB-MULTI) – the social network datasets from TU dataset benchmark [21]. COLLAB dataset consists of ego-networks extracted from public scientific collaboration networks, namely High Energy Physics, Condensed Matter Physics, and Astro Physics. IMDB-B and IMDB-M are movie collaboration ego-networks where a node represents an actor/actress. The statistics of social network datasets are provided in Table 7.

Table 7: Statistics of social network datasets used in link prediction experiments.

Dataset	Graphs	Avg. Nodes	Avg. Edges	Pert. Strength
COLLAB	4320	76.12	2331.37	0.1%
IMDB-B	2039	20.13	85.48	1%
IMDB-M	1478	16.64	77.90	1%

Strategy for Selecting Edges for Perturbations To capture the fine-grained local semantics, we suggest that the weaker magnitude of perturbation is the better (See Section B.6 verifying the effect of edge perturbation strengths). Therefore, we only perturb the tiny amount of edges (e.g., 1 or 2 edges), as shown in Table 7, rightmost column.

Implementation Details We use the Graph Convolutional Network (GCN) [17] consisting of three layers with 300 hidden dimensions. Following the previous works [47, 48], we let the node attributes correspond to the degree of the node. For pre-training, we remove the complete graphs – that always have the edges between any two nodes – as we cannot include additional edges during perturbation. For node masking used in AttrMaksing and our D-SLA, we replace the node attribute with the masked token. For hyperparameters, we use the learning rate of 0.001, the batch size of 32, and the λ_1 of 0.7. During pre-training of our D-SLA, we generate three perturbed graphs by increasing the perturbation magnitudes (e.g., 1%, 2%, 3%). Also, we further mask 20% of nodes in perturbation.

A.4 Analysis

Rank Correlation Coefficient The spearman's rank correlation coefficient measures the correlation between two rank series in the range from -1 to 1, where the value is 1 if the two rank series are perfectly and monotonically the same. We build the following two rank series to compare: 1) the labeled similarity rank between the original and perturbed graphs using the graph edit distance, and 2) the predicted similarity rank based on the embedding-level distances between original and perturbed graphs from pre-trained models. Specifically, we perturb edges of the entire graph by gradually increasing the magnitude of edge perturbation (i.e., 5%, 10%, 15%, 25%, 35%, 45%, 60%, 75%, 90%), and then label ranks of the perturbed graphs to the original graph according to the graph edit distance. Then, the original and perturbed graphs are fed into the pre-trained models, and the ranks are measured by the embedding-level distance between the original and perturbed graphs. Therefore, if a pre-trained model can capture the exact amount of discrepancy, the rank correlation coefficient would be 1, by locating the embedding of a similar graph (a weakly perturbed graph) closer to the original graph than the embedding of a dissimilar graph (a strongly perturbed graph). We measure the coefficient with randomly sampled $1,000$ different graphs.

For models to calculate the similarity across different graphs, we use the pre-trained model for the graph classification task in our D-SLA. For EdgePred, AttrMasking, ContextPred, Informax, and GraphCL baselines, we use the publicly available pre-trained models[5]. For JOAO and GraphLoG, we use the public source codes[6], to obtain the pre-trained models. The WL algorithm in Figure 5 of the main paper corresponds to a randomly initialized GIN model. In other words, since the GIN is as powerful as the WL test, we denote it as the WL algorithm. We evaluate the above models on two different datasets: ZINC15 [29] and QM9 [21], where statistics of each dataset is provided in Table 6. Note that the QM9 dataset is not used for pre-training, thus we can measure the model's generalization ability with it.

Embedding Visualization To visualize the representation space of the original, perturbed, and negative graphs, we pre-train the models on the subset of the ZINC15 dataset [29] with the perturbation strategy in Section A.2 for obtaining perturbed graphs. Then, after pre-training, we visualize the graph representations by PCA [15] and t-SNE [34] for Figure 1 and Figure 3 of the main paper, respectively.

[4] https://chrsmrrs.github.io/datasets/docs/datasets/

[5] https://github.com/snap-stanford/pretrain-gnns, https://github.com/Shen-Lab/GraphCL

[6] https://github.com/Shen-Lab/GraphCL_Automated, https://github.com/DeepGraphLearning/GraphLoG

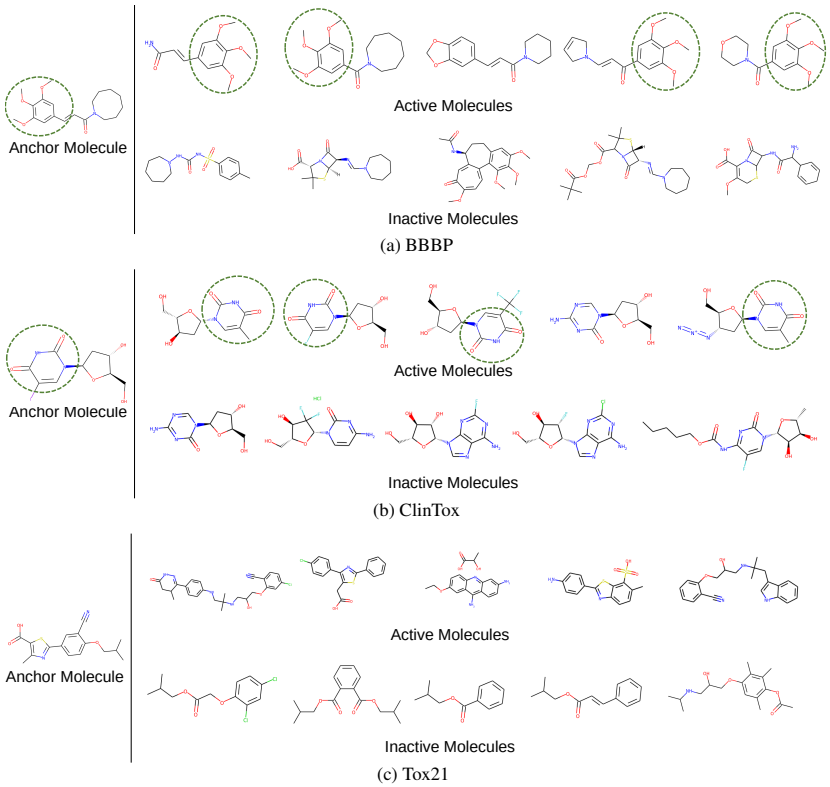


Figure 7: Structural comparisons on the most similar (i.e., top-5) active/inactive molecules for the certain anchor active molecule on the left side, for the BBBP, ClinTox, and Tox21 datasets. Green dotted circles indicate the shared structure across different molecules.

B Additional Experimental Results

In this section, we provide additional results with their corresponding discussions. To be specific, in Section B.1, we analyze the correlation between the characteristics of the dataset and self-supervised learning methods. Then, we provide an in-depth discussion about our observations in the ablation study in Section B.2. Additionally, we provide some guidelines for choosing our hyperparameters (λ_1, α, λ_2, and the perturbation magnitude) in Section B.3, B.4, B.5, and B.6, respectively. Futhermore, we provide an ablation study of attribute masking in Section B.7. Finally, we compare our D-SLA with augmentation-free approaches in Section B.8.

B.1 Dataset Analysis

In this subsection, we further discuss the characteristics of graph self-supervised learning methods with respect to the characteristics of datasets. As shown in Section 4.1, we find that contrastive learning methods outperform predictive learning methods on BBBP and ClinTox. Contrarily, predictive learning methods outperform contrastive learning methods on Tox21. Therefore, we further analyze BBBP, ClinTox, and Tox21 datasets to answer why such methods have counterfactual effects on different datasets.

In Figure 7, we visualize the structures of active/inactive molecules from the anchor molecule. We observe that in BBBP and ClinTox datasets, the activities are highly correlated to the structural similarity, i.e., the structurally similar molecules show the same activities. Therefore, as contrastive learning aims to maximize the similarity between perturbed graphs from the original graph, it fits into the BBBP and ClinTox datasets, showing better performance than predictive learning methods. However, in the Tox21 dataset, we cannot observe any clues that the activities are correlated to the structural similarity. Therefore, capturing the structural similarity with contrastive learning seems to be useless in this dataset, resulting in the better performance of predictive learning methods. However, our D-SLA can learn the discrete embedding space by learning the discrepancy even between similar graphs, thus obtaining a discriminative space that can further be utilized to distinguish between them for downstream tasks and outperforming all other baselines as shown in Section 4.1.

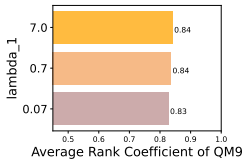
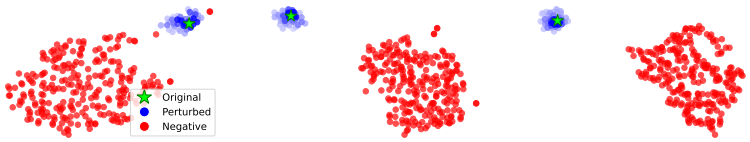


Figure 8: Rank correlation coefficient of QM9 with varying λ_1 values.



(a) $\lambda_1 = 0.07$ (b) $\lambda_1 = 0.7$ (c) $\lambda_1 = 7.0$

Figure 9: Visualization of the learned latent representation space for different λ_1 values. Note that each model is trained on the subset of the ZINC15 dataset and the embedding spaces are visualized by t-SNE [34].

B.2 Additional Interpretation of Ablation Study

We conduct an ablation study on link prediction and graph classification tasks in Table 5 of the main paper. For link prediction, we observe that both two components, $\mathcal{L}_{GD}$ and $\mathcal{L}_{edit}$, consistently improve the performance, thus verifying that our discrepancy-based learning allows the model to capture the local semantics of graphs. Also, for graph classification, we choose the most different two datasets in their properties – ClinTox and BACE datasets – to obviously see the contribution of each component in our D-SLA. In particular, for the ClinTox dataset in which the biochemical activities of molecules are highly correlated to their structures, we observe that it is important to discriminate the negative graphs from the perturbed graphs with the triplet margin loss $\mathcal{L}_{margin}$, as the performance improvements on using it is significant compared to the other dataset: BACE. However, in the case of the BACE dataset, since the molecules are highly similar regardless of their biochemical activities, the graph edit distance loss $\mathcal{L}_{edit}$ largely contributes to the performance gain, allowing the model to learn the exact discrepancy across similar graphs.

B.3 Effect of $\mathcal{L}_{edit}$ Coefficient (λ_1)

We demonstrate the effect of the coefficient λ_1 for $\mathcal{L}_{edit}$ (Equation 8). Specifically, we show its efficacy by measuring the rank correlation coefficient, and by visualizing the graph representation space. Firstly, as shown in Figure 8, the differences in rank correlation with varying scaling coefficient λ_1 are marginal. Also, as shown in Figure 9, for all models, the perturbed graphs are well embedded along their perturbation magnitudes regardless of λ_1 values. Therefore, we suggest that, fortunately, learning the accurate amount of discrepancy between graphs does not heavily depend on the scaling hyperparameter for $\mathcal{L}_{edit}$, namely λ_1. On the other hand, we observe that, if λ_1 is relatively small, some negative graphs are embedded closer to the perturbed ones (Figure 9 (a)). This result indicates that, the regularization effects of our objective $\mathcal{L}_{edit}$ for learning the discrepancy between the original and perturbed graphs can also affect the boundary between perturbed and negative graphs, as shown in Figure 9 (b), (c).

B.4 Effect of Margin (α) in $\mathcal{L}_{margin}$

To demonstrate the effect of the margin α in $\mathcal{L}_{margin}$ (Equation 7), we pre-train the model with varying α values and fine-tune it on BACE dataset for graph classification. As shown in Table 8, large value of α degenerates our discrepancy learning among similar graphs. As described in Section 3.4, if the distance between the original and its negative graph (d'_j) is larger than the distance between the original and perturbed graph (d_i) plus α (i.e., $\alpha + d_i < d'_j$), the distance between the original and perturbed graph (d_i) is preserved not losing the discrepancy learned by $\mathcal{L}_{edit}$ (Equation 6). However, too large α makes it hard to satisfy the condition (i.e., $\alpha + d_i < d'_j$) forcing the model to lose the discrepancy learned by $\mathcal{L}_{edit}$.

Table 8: Effect of α on BACE dataset.

α	ROC-AUC
1.0	83.75 ± 0.96
5.0	83.81 ± 1.01
10.0	78.34 ± 1.07

B.5 Effect of $\mathcal{L}_{margin}$ Coefficient (λ_2)

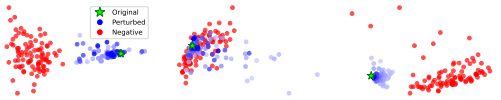
We demonstrate the effect of the coefficient λ_2 for $\mathcal{L}_{margin}$ (Equation 3) by pre-training with various λ_2 values and fine-tuning on BACE dataset for graph classification. As shown in Table 9, when λ_2 is large, the performance on BACE downstream dataset is generated, indicating that the model cannot learn the discrepancy among similar graphs. We suggest that this is because λ_2 controls the intensity of attracting the similar graphs of $\mathcal{L}_{margin}$ and a large value of λ_2 which would strongly attract the similar graphs forces the model to lose the discrepancy among similar graphs learned by $\mathcal{L}_{edit}$ (Equation 6).

Table 9: Effect of λ_2 on BACE dataset.

λ_2	ROC-AUC
0.1	83.68 ± 0.78
0.5	83.81 ± 1.01
0.9	80.72 ± 0.71

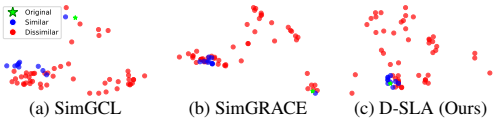
Table 12: Fine-tuning results on graph classification tasks. Best performances are highlighted in bold.

SSL methods	BBBP	ClinTox	MUV	HIV	BACE	SIDER	Tox21	ToxCast	Avg.
SimGCL [49]	67.37 ± 1.23	55.66 ± 4.72	71.24 ± 1.79	75.04 ± 0.86	74.11 ± 2.74	57.44 ± 1.74	74.39 ± 0.45	62.27 ± 0.38	67.19
SimGRACE [40]	71.25 ± 0.86	64.16 ± 4.50	71.18 ± 3.40	74.52 ± 1.12	73.81 ± 1.37	**60.59** ± 0.96	74.20 ± 0.64	63.36 ± 0.52	69.13
D-SLA (Ours)	**72.60** ± 0.79	**80.17** ± 1.50	**76.64** ± 0.91	**78.59** ± 0.44	**83.81** ± 1.01	60.22 ± 1.13	**76.81** ± 0.52	**64.24** ± 0.50	**74.51**



(a) SimGCL (b) SimGRACE (c) D-SLA (Ours)

Figure 10: Embedding space visualization on similar and dissimilar graphs with Graph Edit Distance.



(a) SimGCL (b) SimGRACE (c) D-SLA (Ours)

Figure 11: Embedding space visualization on similar and dissimilar graphs with Tanimoto Similarity.

B.6 Effect of Perturbation Magnitude on Link Prediction

We validate the effect of the perturbation magnitude on the COLLAB dataset for link prediction. As shown in Table 10, we observe that the performance of link prediction is enhanced when only a small amount of edges are perturbed, demonstrating that weaker perturbation magnitude is better for capturing local semantics. If the perturbation magnitude is weak, the perturbed graphs are slightly different from the original graph, thus the model could capture a subtle difference across original and perturbed graphs.

Table 10: Effect of magnitude of perturbation on the link prediction task.

Magnitude	Accuracy
10%	70.67 ± 0.63
1%	74.55 ± 0.76
0.1%	76.19 ± 0.50

B.7 Ablation Study of Attribute Masking

We conduct an additional ablation study for the attribute masking in our perturbation strategy on the COLLAB dataset for link prediction. As shown in Table 11, the performance without attribute masking is significantly lower than the performance with attribute masking. We suggest that, in the pre-training stage, attribute masking limits the information given to the model and forces the model to learn more transferable and fruitful representations, demonstrating that attribute masking in our perturbation is a key factor to learn the local semantics.

Table 11: Ablation study of attribute masking on the link prediction task.

	Accuracy
w/ Masking	76.19 ± 0.50
w/o Masking	70.42 ± 0.95

B.8 Comparison with Augmentation-Free Contrastive Learning Approaches

Recently, augmentation-free contrastive learning methods have been proposed. We validate the effectiveness of our discrepancy learning framework by comparing with augmentation-free approaches on graph classification and link prediction tasks. We compare our D-SLA with SimGCL [49] and SimGRACE [40] that augment views of graphs by adding noise to graph embeddings or model parameters while preserving the graph structures. As shown in Table 12 and Table 13, our discrepancy learning outperforms the augmentation-free contrastive learning approaches, demonstrating the effectiveness of our discrepancy learning framework in capturing both local and global semantics. We further visualize the embedding space of similar and dissimilar graphs with different distance metrics such as Graph Edit Distance (Figure 10) and Tanimoto similarity (Figure 11). We observe that, in SimGCL and SimGRACE, the similar and dissimilar graphs are not distinguished and the augmentation-free approaches cannot capture the exact amount of discrepancy, since they cannot learn the difference between similar graphs. Contrarily, by our discrepancy learning, the model can distinguish similar and dissimilar graphs and learn the exact amount of discrepancy both on Graph Edit Distance and Tanimoto Similarity.

Table 13: Fine-tuning results on link prediction tasks. Best performances are highlighted in bold.

	COLLAB	IMDB-B	IMDB-M	Avg.
SimGCL	77.46 ± 0.86	64.91 ± 2.60	63.78 ± 2.28	68.72
SimGRACE	74.51 ± 1.54	64.49 ± 2.79	62.81 ± 2.32	67.27
D-SLA (Ours)	**86.21** ± 0.38	**78.54** ± 2.79	**69.45** ± 2.29	**78.07**