A Experiments

A.1 Datasets and downstream tasks

Pre-training datasets. AP_NF is collected from the Deep Graph Library package [38] with 73832 nodes in total. It is composed of "computer", "photo" datasets from dgl.data.AmazonCoBuy and "cs", "physics" datasets from dgl.data.Coauthor. SocL_NF is collected from TUDataset [20] with 156754 graphs in total, which is composed of REDDIT-MULTI-12K, dblp_ct1, dblp_ct2, facebook_ct1, facebook_ct2, github_stargazers, highschool_ct1, highschool_ct2, infectious_ct1, infectious_ct2, tumblr_ct1, tumblr_ct2 and twitch_egos. SocS_NF is also collected from TUDataset [20] with 14500 graphs in total, which is composed of IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, REDDITMULTI-5K and COLLAB[4]. MolD is composed of 2000000 unlabeled molecular graphs sampled from ZINC15 [32], the same pre-training dataset used in [15]. The pre-training datasets are also summarized in Table 9.

Downstream datasets. For downstream evaluation on molecular graphs, we use 7 benchmark datasets from MoleculeNet [39][5]. Details of such datasets are presented in Table 10. For dataset split, we adopt the scaffold splitting [3] with the ratio for train/validation/test as 8:1:1. It is a more realistic method for molecular property prediction compared with random splitting and is also the one used in [15, 42]. For downstream evaluation on social graph datasets, we use 4 datasets[6] from Yanardag and Vishwanathan [41]. Details about them are summarized in Table 11. As for dataset splitting method, for each dataset we first split it into train/test sets with the ratio 9:1 and then split the train set into train/validation sets with the ratio 8:1. The validation set is used for model selection. Note that it is different from the splitting method used in GCC[26], where the dataset is randomly split into train/test sets with the ratio 9:1. For downstream node classification datasets, we obtain them (i.e., US-Airport and H-index) from the download link[7] for the downstream datasets provided by the author of [26]. Three different versions of the dataset H-index are provided by the author, among which we use the one named "rand20intop200_5000", which is the same version with the one used in their evaluation process [26]. The way to split those datasets is also kept the same with the one used in [26] (i.e., split into train/test with the ratio 9:1 randomly).

A.2 Implementation Details

A.2.1 Pre-training Configuration

For the fair comparison with other baselines, We use the Graph Isomorphism Network (GIN) [40] with 5 layers and 300 hidden units each layer as our backbones for models pre-trained on all those datasets mentioned above except for AP_NF(whose settings are kept the same with GCC [26]), and mean-pooling to get graph-level representations following [15].

All the pre-training experiments are conducted on a CentOS server equipped with two Intel(R) Xeon(R) Gold 5120 CPU (2.20GHz) and 504G RAM and 8 NVIDIA 32510MiB GPUs. All models are implemented by PyTorch [22] version 1.4.0, DGL [37] with CUDA version 10.1, PyTorch Geometric [9] version 1.4.3, RDKit [8] version 2020.03.2, scikit-learn version 0.22.1 and Python 3.6.10. Information of other packages (like torch_scatter) is presented with the code provided.

For SocL_NF and SocS_NF pre-training, we train for 171465 steps with 32 graphs in each batch. For AP_NF pre-training, we train for 230800 steps with 32 RWR induced subgraphs in each batch and keep other pre-training settings the same with those used in GCC (Moco) pre-training process stated in [26]. For MolD pre-training, HGC and HGC_AdaM have two versions using those two sampling strategies stated in Section 4.2 respectively (first-order neighbourhood sampling termed by FO and high-order graph sampling termed by HO). Details of the hyper-parameters for different strategies are listed in Table 6 (for models using GIN as their backbones) and Table 7 (for models using GCN or GraphSAGE as their backbones). Results for time consumption comparison between HGC, AdaM

[4] All datasets from TUDataset can be downloaded from https://chrsmrrs.github.io/datasets/docs/home/.

[5] All of those datasets can be downloaded from http://moleculenet.ai/datasets-1.

[6] All of them can be downloaded from https://chrsmrrs.github.io/datasets/docs/datasets/.

[7] https://drive.google.com/open?id=12kmPV3XjVufxbIVNx5BQr-CFM9SmaFvM

Table 6: Detailed hyper-parameter settings in pre-training stage for models with GIN as their backbones on the molecular dataset. For abbreviations used, "lr" denotes "learning rate"; "PS" denotes "Positive samples"; "NS" denotes "negative samples"; HA is short for HGC_AdaM. H is short for HGC.

	AdaM	H (FO)	H (HO)	HA (FO)	HA (HO)
Batch size	256	256	256	256	256
Temperature τ	-	0.07	0.07	0.07	0.07
Training steps	781300	156260	156260	156260	156260
Warmup steps	78130	15626	15626	15626	15626
Initial lr	0.001	0.001	0.001	0.001	0.001
#GNN layers	5	5	5	5	5
GNN Hidden size	300	300	300	300	300
Weight decay	0	0	0	0	0
Adam β_1	0.9	0.9	0.9	0.9	0.9
Adam β_2	0.999	0.999	0.999	0.999	0.999
Gradient clipping	1.0	1.0	1.0	1.0	1.0
Dropout rate	0.0	0.0	0.0	0.0	0.0
Walk length	-	1	2	1	2
#Walks	-	1	5	1	5
Mask ratio	0.15	-	-	0.15	0.15
Mask times	5	-	-	3	5
#PS	-	3	5	3	5
#NS	-	255	255	255	255

Table 7: Detailed hyper-parameter settings in pre-training stage for models with GCN or GraphSAGE as their backbones on the molecular dataset. Only those which are different from hyper-parameter settings of models using GIN as their backbones are presented.

	AdaM	H (FO)	H (HO)	HA (FO)	HA (HO)
Walk length	-	1	4	1	2
#Walks	-	1	7	1	5

Table 8: Hyper-parameter in fine-tuning stage and their search space.

Hyper-parameter	Range
Learning rate	0.0001$\sim$0.01
Batch size	32,64,128,256
Dropout ratio	0,0.1,0.2,0.3,0.4,0.5,0.6,0.7
Learning rate scale	0.7,0.8,0.9,1.0,1.1,1.2,1.3
Graph pooling method	mean, sum
Feature combination method	last
#Training epochs	100

and basic pre-training strategies are presented with the code provided (see "README.md" file under the "supp_code" folder).

A.2.2 Fine-tuning Configuration

Fine-tuning evaluation process details. For node classification tasks using the model pre-trained on AP_NF, we adopt the same settings stated in [26] for a fair comparison (i.e. Adam [18] optimizer with learning rate 0.005, learning rate warms up over the first 3 epochs, and linearly decays after 3 epochs). Micro F1-scores on test set after 100 training epochs are reported. For molecular graph classification tasks and social network graph classification tasks, we apply a linear classification layer on top of the pre-trained model, taking pooled graph representations (mean-pooling in our models) as input and output the graph class prediction results. we finetune the pre-trained model for 100 epochs on the train set and report the ROC-AUC (for molecular graph) and micro F1-score

Table 9: Detailed information of pre-training datasets.

Dataset	#Graphs /Nodes	Data Sources
AP_NF	73832	("computers" and "photo") from dgl.data.AmazonCoBuy, ("cs" and "physics") from dgl.data.Coauthor
SocL_NF	156754	REDDIT-MULTI-12K, dblp_ct1&2, facebook_ct1&2, github_stargazers, highschool_ct1&2, infectious_ct1&2, tumblr_ct1&2, twitch_egos
SocS_NF	14500	IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, REDDITMULTI-5K, COLLAB
MolD	2000000	unlabeled molecules sampled from ZINC15 [32]

Table 10: Detailed information of molecular graph downstream dataset.

Dataset	#Tasks	#Compounds
SIDER	27	1427
ClinTox	2	1478
BACE	1	1513
HIV	1	41127
BBBP	1	2039
Tox21	12	7831
ToxCast	617	8575

Table 11: Detailed information of downstream social graph datasets.

Dataset	#graphs	#classes
IMDB-B (IMDB-BINARY)	1000	2
IMDB-M (IMDB-MULTI)	1500	3
RDT-B (REDDIT-BINARY)	2000	2
RDT-M (REDDIT-MULTI-5K)	5000	5

Table 12: Detailed experimental results for different models on molecular datasets. The numbers in brackets are the values of standard deviations.

Backbone	Strategy	**SIDER**	**ClinTox**	**BACE**	**HIV**	**BBBP**	**Tox21**	**ToxCast**
GIN	HGC(FO)	$0.6333_{(0.0121)}$	$0.7919_{(0.0408)}$	$0.8442_{(0.0138)}$	$0.7853_{(0.0072)}$	$0.7217_{(0.0042)}$	$0.7770_{(0.0022)}$	$0.6520_{(0.0052)}$
	HGC(HO)	$0.6237_{(0.0077)}$	$0.8134_{(0.0115)}$	$0.7982_{(0.0201)}$	$0.7687_{(0.0058)}$	$0.7200_{(0.0082)}$	$0.7622_{(0.0021)}$	$0.6379_{(0.0066)}$
	HGC_AdaM(FO)	$0.6118_{(0.0110)}$	$0.7845_{(0.0499)}$	$0.8428_{(0.0064)}$	$0.7839_{(0.0073)}$	$0.7118_{(0.0082)}$	$0.7692_{(0.0030)}$	$0.6537_{(0.0030)}$
	HGC_AdaM(HO)	$0.6183_{(0.0063)}$	$0.7281_{(0.0052)}$	$0.7927_{(0.0187)}$	$0.7672_{(0.0113)}$	$0.7172_{(0.0052)}$	$0.7635_{(0.0025)}$	$0.6459_{(0.0038)}$
GCN	HGC(FO)	$0.6117_{(0.0042)}$	$0.8638_{(0.0051)}$	$0.8405_{(0.0006)}$	$0.7724_{(0.0206)}$	$0.7047_{(0.0031)}$	$0.7581_{(0.0026)}$	$0.6490_{(0.0024)}$
	HGC(HO)	$0.6243_{(0.0044)}$	$0.8359_{(0.0295)}$	$0.8000_{(0.0065)}$	$0.7700_{(0.0029)}$	$0.7168_{(0.0014)}$	$0.7552_{(0.0033)}$	$0.6421_{(0.0022)}$
	HGC_AdaM(FO)	$0.6164_{(0.0103)}$	$0.8231_{(0.0325)}$	$0.8083_{(0.0072)}$	$0.7946_{(0.0102)}$	$0.7189_{(0.0103)}$	$0.7636_{(0.0070)}$	$0.6525_{(0.0025)}$
	HGC_AdaM(HO)	$0.6108_{(0.0037)}$	$0.7801_{(0.0313)}$	$0.8249_{(0.0059)}$	$0.7701_{(0.0060)}$	$0.7006_{(0.0021)}$	$0.7601_{(0.0017)}$	$0.6426_{(0.0021)}$
GraphSAGE	HGC(FO)	$0.6286_{(0.0016)}$	$0.7395_{(0.0284)}$	$0.8368_{(0.0008)}$	$0.7583_{(0.0074)}$	$0.7100_{(0.0016)}$	$0.7575_{(0.0014)}$	$0.6505_{(0.0004)}$
	HGC(HO)	$0.6130_{(0.0089)}$	$0.6242_{(0.0466)}$	$0.7321_{(0.0084)}$	$0.7722_{(0.0149)}$	$0.7129_{(0.0153)}$	$0.7583_{(0.0012)}$	$0.6379_{(0.0066)}$
	HGC_AdaM(FO)	$0.6115_{(0.0040)}$	$0.7164_{(0.0231)}$	$0.7741_{(0.0061)}$	$0.7708_{(0.0053)}$	$0.6951_{(0.0287)}$	$0.7379_{(0.0062)}$	$0.6423_{(0.0009)}$
	HGC_AdaM(HO)	$0.6250_{(0.0029)}$	$0.8127_{(0.0213)}$	$0.7812_{(0.0038)}$	$0.7661_{(0.0085)}$	$0.7187_{(0.0019)}$	$0.7610_{(0.0008)}$	$0.6442_{(0.0018)}$

(for social graph) on the test set at the best validation epoch. We apply three independent randomly initialized runs on each dataset and report the mean (and also standard deviation for molecular graph classification). For social network classification tasks (whether the model is trained on molecular graph dataset presented in Table 5 or social network graph dataset presented in both Table 5 and Table 2), we use the Adam optimizer with learning rate 0.001, weight decay 0, $\beta_1 = 0.9, \beta_2 = 0.999$, batch size 32 and learning rate scale for the linear classification layer 1.0. No hyper-parameter search

is performed[8]. We finetune the pre-trained model for 100 epochs on the train set and report the micro F1-score on the test set at the best validation epoch. For hyper-parameters in the molecular graph fine-tuning process, we use PBT [16] algorithm to search the best combination on the prediction ROC-AUC on the validation set. Details of hyper-parameter search process will be discussed later.

Software and hardware configuration Versions of software used in fine-tuning stage are slightly different from those used in the pre-training stage, i.e., Python 3.7.6, Pytorch 1.4.0, DGL 0.5.3, scikit-learn 0.22.1 for the model trained on AP_NF fine-tuning and Python 3.6.8, Pytorch 1.5.0, Pytorch Geometric 1.6.1, scikit-learn 0.23.2 for fine-tuning other models.

All the fine-tuning experiments are run on a single P40 GPU.

Hyper-parameter selection. For each molecular graph classification task, we use PBT [16] algorithm to search for the best hyper-parameter combination on the prediction ROC-AUC on the validation set with the initial trail number set to 10, parallel trail number 10, maximum trail number 400, mix range 0.3, niche σ 0.1, niche α 1.0, perturb factor 0.0001 for continuous parameter (i.e., learning rate). Table 8 shows all the hyper-parameters and their search space in the fine-tuning stage. Search results can be found together with the code provided (see "README.md" file under the "supp_code" folder for further instructions).

A.3 Baselines

A.3.1 Molecular Graph Classification

Hu. et al. [15] Four of the pre-training strategies from Table 1 (i.e., Edge_Pred, Infomax, Attr_Mask, Context_Pred) are proposed in [15]. We download the author's code and pre-trained models that are released officially and apply three independent runs on each downstream fine-tuning dataset using hyper-parameter combinations provided by the authors (i.e., learning rate = 0.001, dropout rate = 0.5, batch size = 32).

Code: https://github.com/snap-stanford/pretrain-gnns

GraphCL [42]. We download the author's code released officially and use the weights of the pre-trained model they provide (i.e., `graphcl_80.pth`). We apply three independent runs on each downstream fine-tuning dataset using the hyper-parameters provided by the authors (i.e., leaning rate = 0.001, dropout rate = 0.5, batch size = 32, number of training epochs = 100, learning rate scale for the linear classification layer = 1.0).

Code: https://github.com/Shen-Lab/GraphCL

C_Subgraph [26]. We download the code of [26] released by the authors officially. Since their implementation cannot be used in both our molecular graph pre-training stage as well as the downstream molecular graph classification stage directly, we carefully implement their graph sampling strategy, keeping the restart probability and the walk length same with their default settings (i.e., restart probability = 0.8, walk length is determined by a fixed number and the node's degree, the difference is that we would perform such RWR process for several times to sample enough nodes due to the different version of DGL we use from GCC (see A.2.1)) and use it to pre-train our model using the same pre-training settings with our HGC pre-training stage. The downstream evaluation process is kept the same with those for our model (on molecular graph classification datasets).

Code: https://github.com/THUDM/GCC

A.3.2 Node Classification & Social Network Graph Classification

All the results of the baselines presented in Table 3 and Table 2 are taken directly from the paper [26], since the test set selection process and the evaluation metric (i.e., micro F1-score) are kept the same with [26] and that we carefully check the author's description and implementation details of the baselines used in their paper and choose to trust in their implementation and evaluation for those baselines.

[8] We take the entire graphs as our graph instances in our evaluation process, different from RWR induced subgraphs in the evaluation process of GCC [26].

A.4 Details about Experimental Setup

A.4.1 Two-step hierarchical graph construction process

We adopt a two-step approaches to construct the hierarchical graph:

- Candidate selection: We first sort graphs in the dataset by molecular weight (calculated by MolWt($\cdot$) function in the software RDKit [8]) for molecular graphs or number of nodes for graphs without node features. For molecular graphs, we select molecule B as A's candidates if and only if: 1). The molecular weights of A and B differ by no more than 10% of A; 2). The number of rings of A and B differs by no more than 1; 3). The number of atoms in A and B differs by no more than 7; 4). The number of candidates have been selected is still less than a manually defined value (70 in our pre-processing). For graphs without node attributes, we select graph A as graph B's candidate if and only if: 1). The number of nodes of A and B differ by no more than 10%. 2). The number of edges of A and B differs no more than 10%. 3). The number of candidates have been selected is still less than a manually defined value (70 in our pre-processing).
- Edge construction: For each graph A in graph B's candidate set, we calculate their similarity score and build an edge between them in the constructing hierarchical graph if their similarity score is above a pre-defined threshold τ.

A.4.2 Details of adaptive masking process

Detailed algorithms. The main algorithm of the adaptive masking (AdaM) process is summarized in Algorithm 1. Details of its sub-algorithm – PScore is demonstrated in Algorithm 2, where MASKNODE$(G, \mathcal{S})$ takes a graph G and a node set for masking $\mathcal{S}$ as input and outputs a graph G' with attributes of nodes in $\mathcal{S}$ masked.

Examples. An example regarding to the adaptive masking process is illustrated in Fig. 3. A toy example mentioned in Sec. 4.4 claiming that the uniform selection strategy may break structural relations among nodes in graphs is: suppose that node v has only one neighbor: node u. In this case, if we mask all features of u and v simultaneously, it is very hard for model $\mathcal{M}$ to make a good prediction of v since v is highly related to u. In other words, $\mathcal{M}$ cannot encode the attribute distribution of v. Therefore, a good masking set should have less correlations with respective to the output of the model $\mathcal{M}$.

Algorithm 1 Adaptive Masking.

Input: Input graph $G(\mathcal{V}, \mathcal{E}, \mathbf{X})$; The model $\mathcal{M}$; Masking steps T; The number of nodes for masking at each step α.
Output: Masked node set $\mathcal{S}$;

$\mathcal{S} \leftarrow \emptyset$
$\mathcal{S}_{prev} \leftarrow \emptyset$
for $t = 1$ to T **do**
  if t == 1 **then**
    Randomly select a node set $\mathcal{K}$ with α nodes from $\mathcal{V}$ with a uniform distribution.
  else
    for $v \in \mathcal{V} \setminus \mathcal{S}$ **do**
      $s_v \leftarrow \text{PScore}(v, \mathcal{M}, G, \mathcal{S}, \mathcal{S}_{prev})$
    Randomly select a node set $\mathcal{K}$ with α nodes form $\mathcal{V} \setminus \mathcal{S}$ with probability for each node v :

$$p_v = \frac{s_v}{\sum_{u \in \mathcal{V} \setminus \mathcal{S}} s_u}$$

  $\mathcal{S}_{prev} \leftarrow \mathcal{S}$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{K}$
return $\mathcal{S}$

Algorithm 2 PScore.

Input: Target node v; The model $\mathcal{M}$; Input graph $G(\mathcal{V}, \mathcal{E}, \mathcal{X})$; Current masked node set $\mathcal{S}_{\text{cur}}$; Previous masked node set $\mathcal{S}_{\text{prev}}$;

$G_{\text{prev}} \leftarrow \text{MASKNODE}(G, \mathcal{S}_{\text{prev}})$;
$G_{\text{cur}} \leftarrow \text{MASKNODE}(G, \mathcal{S}_{\text{cur}})$;
$\boldsymbol{y}_{v,\text{prev}} \leftarrow \mathcal{M}(v, G_{\text{prev}})$;
$\boldsymbol{y}_{v,\text{cur}} \leftarrow \mathcal{M}(v, G_{\text{cur}})$;
$s \leftarrow 1 - \text{cross\_entropy}(\boldsymbol{y}_{v,\text{prev}}, \boldsymbol{y}_{v,\text{cur}})$;
return s;

A.4.3 Time Consumption for Pre-processing

No more than 4 hours for MolD and about 2 hours for SocL_NF, details are presented with code provided (see "README.md" file under the "supp_code" folder).

A.5 Additional Experimental Results

A.5.1 More experimental results for the influence of data augmentations on the quality of positive instances

We present more experimental results w.r.t. how the quality of resulting positive instances is influenced by data augmentation methods applied. This part is to support the claim made in Sec. 1 that popular data augmentation strategies cannot get positive graph instances with ideal properties preserved for various kinds of graph data in addition to Fig. 1.

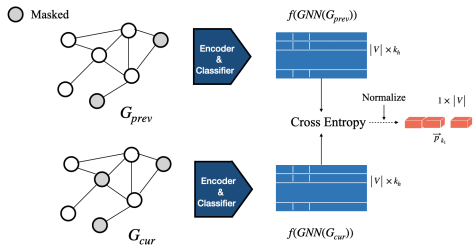

Figure 3: A toy example to illustrate how we determine the masking probability in each masking step (i.e., k_1-th), assuming that $k_1 > 2$.

The claim that the constructed similarity based hierarchical graph encodes the similarity hierarchy is reasonable and can be supported by statistical results for average similarity scores between the target graph instances and their neighbouring graph instances in different hops. As shown in Fig. 4, the average fingerprint similarity score between molecules in different hops and the target graph instance decreases as the hop increases. Moreover, the deceasing speed is relatively low compared with the decreasing speed of the average similarity scores between the positive instances obtained by graph data augmentation strategies and the target graph instance w.r.t. the augmentation ratio (e.g. Fig. 1).

Analysis for graph sampling (subgraph) strategy on molecular graphs. Graph sampling data augmentation strategy samples subgraphs as positive instances for the target graph instance. It is an effective strategy for graphs with no node/edge attributes [26]. Admittedly, perhaps it can also get subgraphs that are structurally similar enough (measured by Weisfeiler-Lehman Graph Kernel normalized similarity) with each other whether for those graphs with node/edge attributes. It can hardly maintain specific domain information that is important for some kinds of graph data (e.g., molecular graphs) or cannot guarantee enough semantic similarity between obtained positive graph instances and the original graph instance. We use the code for subgraph sampling provided by [42] to investigate further. We discover that 1). subgraphs cannot preserve enough semantic similarity (measured by fingerprint similarity scores) between obtained positive instances and the target graph instance for molecular graphs; 2). sampled subgraphs may not be proper molecules anymore. Statistical results calculated on 1000 molecules randomly sampled from our molecular pre-training dataset when the subgraph sampling ratio is relatively low (≤ 0.25) and relatively high (up to 0.8) are presented in Table 13 and Table 14 respectively. Perhaps it is surprising that the fingerprint similarity scores between two sampled subgraphs still remain at a low level even when the data augmentation ratio is relatively high (i.e. up to 0.8).

Analysis for attribute masking strategy on molecular graphs. We also observe similar phenomenon for attribute masking strategy when applied on molecules. As shown in Table 15, the

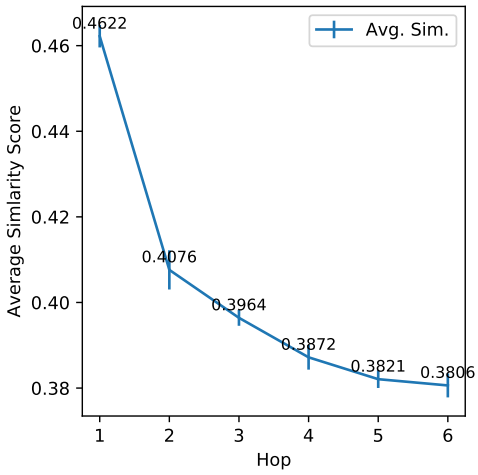

Figure 4: The fingerprint similarity scores between graph instances in the hierarchical graph and the target graph instance w.r.t. the number of hops. Calculated on 1000 randomly selected molecules from MolD. Three independent runs were given, with average similarity scores and standard deviation reported.

Table 13: Statistical results for the ratio of sampled subgraphs to be proper molecules and their similarity scores with the target molecule w.r.t. subgraph sampling ratio. Calculated on 1000 molecules randomly sampled from MolD. Three independent runs were given, with mean and standard deviation values reported. This table presents results when the subgraph sampling ratio is relatively low. Values presented in the table have the format mean $\pm$ std.

	0.10	**0.15**	**0.20**	**0.25**
Ratio to be proper molecules	0.511 ± 0.011	0.456 ± 0.005	0.477 ± 0.034	0.522 ± 0.028
Similarity scores	0.00239 ± 0.00008	0.00292 ± 0.00007	0.00296 ± 0.00010	0.00235 ± 0.00003

average similarity score between the resulted positive graph instances and their original graph instance drops as the attribute masking ratio increases, which is not a surprising phenomenon. However, unlike the graph sampling strategy, attribute masking strategy always fail to get masked graphs that are legal molecules, perhaps due to the loss of node attributes which are important for molecules.

Analysis for the effectiveness of graph data augmentation strategies on social network graphs. We also investigate into how common graph data augmentation strategies will influence the resulting positive instances' similarity scores with the target graph instances when applied on social network graphs. The results are shown in Fig. 5 for dropping nodes and dropping edge strategies and Fig. 6 for subgraph augmentation strategy. The experiment is conducted on 1000 graph instances uniformly randomly chosen from our SocL_NF dataset. Three independent experiments are performed with the average and standard deviation value of the average similarity score over such 1000 graph instances in each run are reported. The similarity score measurement is Weisfeiler-Lehman Graph Kernel normalized similarity provided by Python Package GraKel [31]. Compared with the average similarity between the target graph instance and its first-order graph instances in the constructed hierarchical graph, which is 0.3467, such two data augmentation strategies cannot get positive graph instances that are similar enough with the target instance even when the data augmentation ratio is relatively low (e.g. 0.1). Moreover, the average similarity score drops obviously as the augmentation ratio increases, which is not a surprising phenomenon.

Table 14: Statistical results for the ratio of sampled subgraphs to be proper molecules and their similarity scores with the target molecule w.r.t. subgraph sampling ratio. Calculated on 1000 molecules randomly sampled from MolD. Three independent runs were given, with mean and standard deviation values reported. This table presents results when the subgraph sampling ratio is relatively high. Values presented in the table have the format mean $\pm$ std.

	0.50	**0.70**	**0.80**
Ratio to be proper molecules	0.511 ± 0.012	0.515 ± 0.001	0.506 ± 0.014
Similarity scores	0.00211 ± 0.00004	0.00219 ± 0.00008	0.00223 ± 0.00001

Table 15: Statistical results for the ratio of the resulting masked graphs to be proper molecules and their similarity scores with their respective original molecules w.r.t. attribute masking ratio. Calculated on 1000 molecules randomly sampled from our pre-training dataset. Three independent runs were given, with mean and standard deviation values reported.

	0.10	**0.15**	**0.20**	**0.25**
Ratio to be proper molecules	0.014 ± 0.002	0.015 ± 0.003	0.014 ± 0.002	0.014 ± 0.003
Similarity scores	0.472 ± 0.004	0.414 ± 0.001	0.378 ± 0.003	0.359 ± 0.002

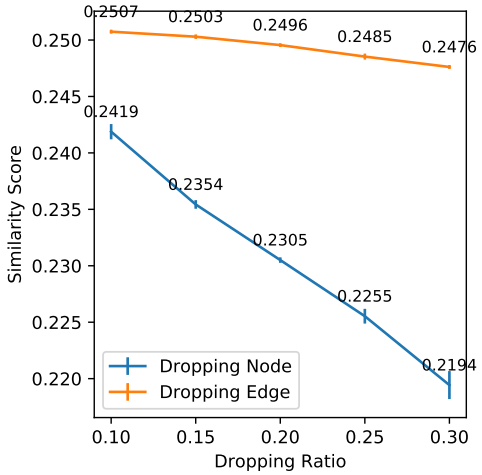

Figure 5: The Weisfeiler-Lehman Graph Kernel normalized similarity scores between positive graph instances, obtained by applying dropping nodes or dropping edges data augmentation strategies, and their respective target graph instances w.r.t. augmentation ratio. Three independent runs were given, with average similarity scores and standard deviation values reported.

As for the subgraph augmentation strategy, it can be seen that the average similarity score lies in a low level for both low and high augmentation ratio. Such phenomenon is similar with the one observed on molecular graphs.

A.6 Additional ablation study of the effectiveness of pre-training strategies on GCN and GraphSAGE

Similar with the effectiveness of our pre-training strategies (HGC and AdaM) for GIN models, we can also observe significant contribution of our pre-training strategies on other backbones (GCN

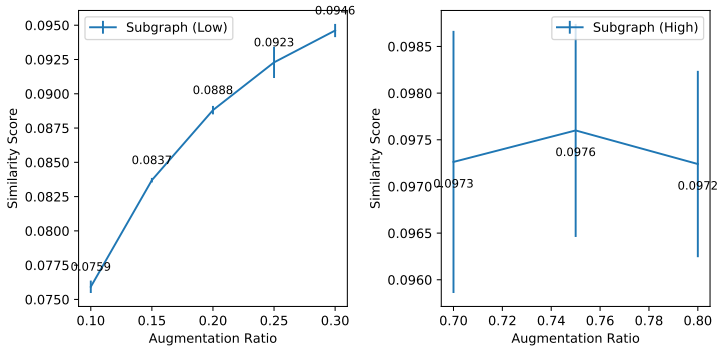

Figure 6: The Weisfeiler-Lehman Graph Kernel normalized similarity scores between positive graph instances, obtained by applying the subgraph data augmentation strategy, and their respective target graph instances w.r.t. subgraph augmentation ratio. Left: changing curve of the average similarity score w.r.t. data augmentation ratio (relatively low). Right: changing curve of the average similarity score w.r.t. data augmentation ratio (relatively high). Three independent runs were given, with average similarity scores and standard deviation values reported.

Table 16: Effectiveness of the pre-training on GCN and GraphSAGE (denoted as "SAGE" in brackets). Bold numbers for absolute improvements larger than 0.05. For abbreviations used, "No-Pret." denotes trained-from-scratch models, "Pret." denotes pre-trained models (presented are the best values achieved by our different pre-training strategies for each downstream evaluation dataset), "Abs. Imp." denotes the absolute ROC-AUC improvement.

	No-Pret. (GCN)	**Pret. (GCN)**	**Abs. Imp. (GCN)**	**No-Pret. (SAGE)**	**Pret. (SAGE)**	**Abs. Imp. (SAGE)**
SIDER	0.6072	0.6243	+0.0171	0.6173	0.6286	+0.0113
ClinTox	0.5866	0.8638	**+0.2772**	0.6936	0.8127	**+0.1191**
BACE	0.7652	0.8405	**+0.0753**	0.7285	0.8368	**+0.1083**
HIV	0.7547	0.7946	+0.0399	0.7477	0.7730	+0.0253
BBBP	0.6758	0.7189	+0.0431	0.6826	0.7187	+0.0361
Tox21	0.7370	0.7636	+0.0266	0.7609	0.7643	+0.0034
ToxCast	0.6394	0.6525	+0.0131	0.6395	0.6505	+0.0110

Table 17: The average minimum distances between nodes masked by the random masking strategy and our adaptive making strategy with different masking times k. Presented are the mean of the values calculated over the first five epochs.

	RdM	$k=3$	$k=5$	$k=6$	$k=7$
Avg. Dis.	2.7519	2.7584	2.7623	2.7625	2.7670

and GraphSAGE) as shown in Table 16. We can arrive at a conclusion, which is similar with the one made in [15], that more powerful backbone can benefit more from pre-training by comparing Table 16 with Table 4. Besides, pre-trained GCN models can get larger benefit on the dataset BACE (27.72% absolute improvement) compared with the improvement made by pre-trained GIN (16.54%) and GraphSAGE (10.83%).

A.7 Additional understanding and analysis for adaptive masking strategy

To evaluate the effectiveness of the proposed adaptive masking strategy, we compare the performance of GIN pre-trained by two different masking strategies (i.e., the uniform random masking strategy, denoted as RdM and the adaptive masking strategy AdaM) on molecular classification datasets and summarize them in Table 18. It can be seen that when using AdaM to select nodes to mask, the mask-and-predict paradigm may probably help the model learn more inherent graph-level molecular properties, thus getting better performance in downstream tasks (e.g., with maximum absolute ROC-AUC increase 11.4% for the dataset ClinTox and 3.80% in average over all of the molecular datasets).

Therefore, we want to ask that what does AdaM bring to us, or what makes it different from the random masking strategy? One intuition is that if nodes that are less disturbed had larger probabilities to be masked in the current masking step, then nodes selected by AdaM will be distributed more evenly in the graph than nodes chosen by RdM. For further investigation, we calculate the average minimum distances between masked nodes[9] when adopting RdM and AdaM with different masking steps (k) over the first five training epochs in Table 17. All masking ratios are set to 15%. Two observations can be made from Table 17: 1). The average minimum distances between masked nodes chosen by AdaM are larger than those of RdM, which confirms our conjecture; 2). The average minimum distance between masked nodes grows as the masking times k increases. Hence, if we apply more masking steps, masked nodes will be more likely to distribute evenly over the molecule.

B Further Analysis for Similarity-aware Sampling Strategy

In this section, we conduct some further analysis of the proposed similarity-aware sampling strategy. This section is to support Sec. 4.3.

B.1 Approximate similarity function

The pre-training graph dataset can be divided into two sets for each graph instance $G_i \in \mathcal{G}$ based on the ground-truth similarity function: $\text{sim}_{\text{gt}}(\cdot,\cdot)$ and a graph instance $G_i \in \mathcal{G}$: $\mathcal{G}_i^{\text{gt}+} =$

[9]For each node i in the masked nodes set $\mathcal{N}_s$, we calculate the minimum distance d_{min} measured by the length of the shortest path between it and other nodes in the set (i.e., $d_{min} = \min\{\text{dis}(i,j)|j \in \mathcal{N}_s, j \neq i\}$). Then the average value over all nodes in the masked nodes set is reported.

Table 18: Effectiveness of dynamic masking strategy v.s. uniformly random masking strategy. Bold numbers for those larger than 0.05.

	RdM	AdaM	**Abs. Imp.**
SIDER	0.5947	0.6164	+0.0217
ClinTox	0.6685	0.7797	**+0.1139**
BACE	0.8064	0.8224	+0.0160
HIV	0.7668	0.7704	+0.0036
BBBP	0.6316	0.7273	**+0.0957**
Tox21	0.7657	0.7696	+0.0039
ToxCast	0.6463	0.6603	+0.0160

$\{G_k|\text{sim}_{\text{gt}}(G_i, G_k) = 1, G_k \in \mathcal{G}\}$, containing G_i's ground-truth positive instances and $\mathcal{G}_i^{\text{gt}-} = \{G_k|\text{sim}_{\text{gt}}(G_i, G_k) = 0, G_k \in \mathcal{G}\}$, containing G_i's negative graph instances.

In practice, we use the approximate similarity function to approximate the ground-truth similarity function. We introduce the definition of such functions together with some reasonable properties assumed for their probability density functions over each $\mathcal{G}_i^{\text{gt}+}$ and $\mathcal{G}_i^{\text{gt}-}$ as follows:

Definition 2 (Approximation similarity function)**.** *For a similarity function sim$(\cdot,\cdot)$ and a graph instance $G_i \in \mathcal{G}$, denote its similarity score distribution density function over G_i's ground-truth positive instance set $\mathcal{G}_i^{gt+}$ as $f_i^+(\cdot)$ and that over $\mathcal{G}_i^{gt-}$ as $f_i^-(\cdot)$. Two properties are assumed for $f_i^+(\cdot)$ and $f_i^-(\cdot)$: 1). Both $f_i^+(\cdot)$ and $f_i^-(\cdot)$ are first-order differentiable functions over their domain of definition $[0, 1]$; 2). There exists a similarity score threshold $0 < x_0 < 1$, s.t. for all $x_0 < x_1 < 1$, we have $\int_{x_1}^1 f_i^+(x)\mathrm{d}x > \int_{x_1}^1 f_i^-(x)\mathrm{d}x$.*

Some specific probability density functions can be found for $f_i^+(\cdot)$ and $f_i^-(\cdot)$ such as normal distribution probability density functions (truncated between $[0, 1]$ and re-normalized by the integral over $[0, 1]$) and beta distribution probability density functions, in which cases the above mentioned properties can be easily satisfied by further restricting the relationship between their parameters.

In our practice, another parameter – a similarity threshold τ is introduced to divide the graph pre-training dataset $\mathcal{G}$ into two sets for each $G_i \in \mathcal{G}$: $\mathcal{G}_i^{\text{sim}_\tau+} = \{G_k|\text{sim}(G_i, G_k) \geq \tau, G_k \in \mathcal{G}\}$ and $\mathcal{G}_i^{\text{sim}_\tau-} = \{G_k|\text{sim}(G_i, G_k) < \tau, G_k \in \mathcal{G}\}$.

B.2 Why we should avoid false-positive instances in the contrastive learning process?

A graph instance G_k is G_i's false-positive instance if and only if $G_k \in G_i^{\text{sim}_\tau+} \wedge G_k \notin \mathcal{G}_i^{\text{gt}+}$. It is intuitively correct that we should avoid such false-positive instances in the contrastive learning process. We investigate into such intuition by analyzing how false-positive instances would influence the contrastive training process in this section.

To get a glimpse into the training process, we start by introducing the loss function which is widely used in the contrastive learning process.

The contrastive learning loss $\mathcal{L}$ is composed of losses from each graph instance in the pre-training dataset:

$$\mathcal{L} = \sum_{G_i \in \mathcal{G}} \mathcal{L}_i. \tag{3}$$

where G_i denotes a graph instance from the pre-training dataset. Moreover, we also use x_j to denote the same instance (with G_j) in G_i's positive sampling probability function $P_i^+(\cdot)$[10] as well as the negative instance sampling probability function $P_i^-(\cdot)$ for simplicity. The actual value of $\mathcal{L}_i$ in each training epoch may be different from each other, since it is determined by the selected positive graph instance G_i^+ and each negative graph instance G_i^-. Thus, we use its expectation value here:

$$\mathbb{E}[\mathcal{L}_i] = -\mathbb{E}_{x_i^+ \sim P_i^+} \log \left[\frac{\exp(z_i^T z_i^+/\tau_t)}{\exp(z_i^T z_i^+/\tau_t) + N\mathbb{E}_{x_i^- \sim P_i^-} \exp(z_i^T z_i^-/\tau_t)} \right], \tag{4}$$

[10]To be more specific, $P_i^+(x_j)$ is the probability of sampling the graph instance G_j as graph instance G_i's positive instance in the contrastive learning process.

where τ_t is the temperature hyper-parameter, N is the number of negative instances sampled for each instance in one training epoch, $z_i = \frac{w_i}{\|w_i\|_2} \in \mathbb{R}^{K\times 1}$ is the normalized representation vector of graph instance G_i with the feature dimension K, w_i is the corresponding representation vector output by the neural network $\|w_i\|_2$ is the 2-norm of w_i. Here, we assume that negative instances are chosen from a uniform distribution over the whole pre-training dataset, which means that $P_i^-(x_j) = \frac{1}{|\mathcal{G}|}$ for each $x_i \in \mathcal{G}$ and $x_j \in \mathcal{G}$. Thus, two partial derivatives of interest are as follows:

$$\mathbb{E}\left[\frac{\partial \mathcal{L}_j}{\partial w_i}\right] = -P_j^+(x_i)\frac{1}{\tau_t\|w_i\|_2}(1 - z_i z_i^T)\left[z_j - \frac{z_j \exp(z_j^T z_i/\tau_t) + \frac{N}{|\mathcal{G}|} z_j \exp(z_j^T z_i/\tau_t)}{\exp(z_j^T z_i/\tau_t) + \frac{N}{|\mathcal{G}|}\sum_{x_j^- \in \mathcal{G}} \exp(z_j^T z_j^-/\tau_t)}\right] \tag{5}$$

$$= -P_j^+(x_i)\frac{1}{\tau_t\|w_i\|_2}(z_j - (z_i^T z_j)z_i)\left[\frac{\frac{N}{|\mathcal{G}|}\sum_{x_j^- \in \mathcal{G}\setminus\{x_i\}} \exp(z_j^T z_j^-/\tau_t)}{\exp(z_j^T z_i/\tau_t) + \frac{N}{|\mathcal{G}|}\sum_{x_j^- \in \mathcal{G}} \exp(z_j^T z_j^-/\tau_t)}\right] \tag{6}$$

Let

$$Q(x_j, x_i) = \frac{\frac{N}{|\mathcal{G}|}\sum_{x_j^- \in \mathcal{G}\setminus\{x_i\}} \exp(z_j^T z_j^-/\tau_t)}{\exp(z_j^T z_i/\tau_t) + \frac{N}{|\mathcal{G}|}\sum_{x_j^- \in \mathcal{G}} \exp(z_j^T z_j^-/\tau_t)} \tag{7}$$

$$P(x_j, x_i) = \frac{\frac{N}{|\mathcal{G}|}\exp(z_j^T z_i/\tau_t)}{\exp(z_j^T z_i/\tau_t) + \frac{N}{|\mathcal{G}|}\sum_{x_j^- \in \mathcal{G}} \exp(z_j^T z_j^-/\tau_t)}, \tag{8}$$

then we have:

$$\mathbb{E}\left[\frac{\partial \mathcal{L}_j}{\partial w_i}\right] = -P_j^+(x_i)\frac{1}{\tau_t\|w_i\|_2}(z_j - (z_i^T z_j)z_i)Q(x_j, x_i) \tag{9}$$

As for $\mathbb{E}\left[\frac{\partial \mathcal{L}_i}{\partial w_i}\right]$, we have:

$$\mathbb{E}\left[\frac{\partial \mathcal{L}_i}{\partial w_i}\right] = -\frac{1}{\tau_t\|w_i\|_2}(1 - z_i z_i^T)\sum_{x_i^+ \in \mathcal{G}_i^+} P_i^+(x_i^+)\left(z_i^+ - \frac{z_i^+ \exp(z_i^T z_i^+/\tau_t) + \frac{N}{|\mathcal{G}|}\sum_{x_i^- \in \mathcal{G}} z_i^- \exp(z_i^T z_i^-/\tau_t)}{\exp(z_i^T z_i^+/\tau_t) + \frac{N}{|\mathcal{G}|}\sum_{x_i^- \in \mathcal{G}} \exp(z_i^T z_i^-/\tau_t)}\right) \tag{10}$$

$$= -\frac{1}{\tau_t\|w_i\|_2}(1 - z_i z_i^T)\sum_{x_i^+ \in \mathcal{G}_i^+} P_i^+(x_i^+)\left(z_i^+(P(x_i, x_i^+) + Q(x_i, x_i^+)) - \sum_{x_i^- \in \mathcal{G}} z_i^- M^-(x_i, x_i^+, x_i^-)\right) \tag{11}$$

$$= -\frac{1}{\tau_t\|w_i\|_2}(1 - z_i z_i^T)\sum_{x_i^+ \in \mathcal{G}_i^+} P_i^+(x_i^+)\left(z_i^+(P(x_i, x_i^+) + Q(x_i, x_i^+))\right) \tag{12}$$

$$+ \frac{1}{\tau_t\|w_i\|_2}(1 - z_i z_i^T)\sum_{x_i^- \in \mathcal{G}} z_i^-\left(\sum_{x_i^+ \in \mathcal{G}_i^+} P_i^+(x_i^+)M^-(x_i, x_i^+, x_i^-)\right) \tag{13}$$

$$= -\frac{1}{\tau_t\|w_i\|_2}(1 - z_i z_i^T)\sum_{x_i^+ \in \mathcal{G}_i^+} P_i^+(x_i^+)\left(z_i^+(P(x_i, x_i^+) + Q(x_i, x_i^+))\right) \tag{14}$$

$$+ \frac{1}{\tau_t\|w_i\|_2}(1 - z_i z_i^T)\sum_{x_i^- \in \mathcal{G}} z_i^- M(x_i, x_i^-), \tag{15}$$

where

$$M^-(x_i, x_i^+, x_j) = \frac{\frac{N}{|\mathcal{G}|}\exp(z_i^T z_j/\tau_t)}{\exp(z_i^T z_i^+/\tau_t) + \frac{N}{|\mathcal{G}|}\sum_{x_i^- \in \mathcal{G}} \exp(z_i^T z_i^-/\tau_t)}, \tag{16}$$

$M(x_i, x_j)$ is the weighted sum of $M^-(x_i, x_i^+, x_j)$, thus is the function of only x_i and x_j:

$$M(x_i, x_j) = \sum_{x_i^+ \in \mathcal{G}_i^+} P_i^+(x_i^+) \frac{\frac{N}{|\mathcal{G}|} \exp(z_i^T z_j / \tau_t)}{\exp(z_i^T z_i^+ / \tau_t) + \frac{N}{|\mathcal{G}|} \sum_{x_i^- \in \mathcal{G}} \exp(z_i^T z_i^- / \tau_t)} \tag{17}$$

$$= \sum_{x_i^+ \in \mathcal{G}_i^+} P_i^+(x_i^+) M^-(x_i, x_i^+, x_j). \tag{18}$$

Then, the expectation value of the partial gradient of the contrastive learning loss $\mathcal{L}$ on G_i's representation vector w_i brought by its positive instance x_j is composed of the following two items:

$$\mathbb{E}\left[\frac{\partial \mathcal{L}_i}{\partial w_i}\right](x_j) = -\frac{1}{\tau_t \|w_i\|_2} P_i^+(x_j)(z_j - (z_i^T z_j) z_i)(Q(x_i, x_j) + P(x_i, x_j)) \tag{19}$$

$$\mathbb{E}\left[\frac{\partial \mathcal{L}_j}{\partial w_i}\right] = -\frac{1}{\tau_t \|w_i\|_2} P_j^+(x_i)(z_j - (z_i^T z_j) z_i) Q(x_j, x_i). \tag{20}$$

Following [17], we have:

$$\|z_j - (z_i^T z_j) z_i\| = \sqrt{1 - (z_i^T z_j)^2}, \tag{21}$$

which indicates that the magnitude of the partial gradient of contrastive learning loss on instance x_i's representation vector w_i from its positive instance x_j is relevant with the cosine similarity $(z_i^T z_j)$ between their representation vectors.

Then, what we wish to explain here is that false-positive instances may have negative impact on the ability of the network to converge to the optimal state, which can be reached if we have the knowledge of each graph instance's ground-truth semantic class in the pre-training dataset.

Let us assume that we have arrived at a training stage where the network has been optimized to a near-optimal state such that the cosine similarity between positive instance pairs' representation vectors are relatively high (i.e., $z_i^T z_i^+ \approx 1$), while those between negative instance pairs are relatively low (i.e., $z_i^T z_i^- \approx 0$). In this stage, the magnitude of the contrastive learning loss's gradient on instance x_i's representation vector w_i from its ground-truth positive instance x_i^+ may be relatively low given that $z_i^T z_i^+ \approx 1$. However, the magnitude of such gradient from its false-positive instance x_j may be relatively high since $z_i^T z_j \approx 0$. Thus, the direction of $\frac{\partial \mathcal{L}}{\partial w_i}$ will be closer to that of $z_j - (z_i^T z_j) z_i$. It may be a bit deviated from the direction of the gradient from its ground-truth positive instance z_i^+ since:

$$(z_i^+ - (z_i^T z_i^+) z_i)^T (z_j - (z_i^T z_j) z_i) = z_i^{+T} z_j - z_i^{+T} z_i z_j^T z_i, \tag{22}$$

which is near to zero since $z_i^T z_j \approx 0$ and $z_i^{+T} z_i \approx 1$.

Such near orthogonal optimization direction implies the incorrect optimization direction and the unstable training process.

B.3 Higher similarity score may indicate higher probability to be a ground-truth positive instance

In this section, we want to investigate into whether it is reasonable to give graph instances that have higher similarity scores with the target graph instance G larger probability to be sampled as its positive instances. We aim to find a similarity score interval such that graph instances that have larger similarity scores with G_i means they also have higher probability to be G_i's ground-truth positive instances when their similarity scores are changing in such an interval.

Assume that we observe an instance G_j whose similarity score with the target instance G_i is x, denote the event that G_j is a ground-truth positive instance of G_i as A and the event that G_j is a negative instance of G_i as B. Then,

$$P(\text{sim}(G_i, G_j) = x | A) = f_i^+(x) \delta x \tag{23}$$

$$P(\text{sim}(G_i, G_j) = x | B) = f_i^-(x) \delta x, \tag{24}$$

where $0 < \delta x \ll 1$ is a small quantity, $\text{sim}(\cdot, \cdot)$ is the similarity score function we use in practice, $f_i^+(\cdot)$ and $f_i^-(\cdot)$ are the corresponding similarity score probability density functions over G_i's ground-truth positive instance set $\mathcal{G}_i^{\text{gt}+}$ and its negative instance set $\mathcal{G}_i^{\text{gt}-}$.

Assume that we have no prior knowledge of the relationship between G_i and G_j, which means that $P(A) = P(B) = \frac{1}{2}$. Then, the posterior probabilities of the occurrence of the event A and B are as follows:

$$P(A|\text{sim}(G_i, G_j) = x) = \frac{f_i^+(x)\delta x}{f_i^+(x)\delta x + f_i^-(x)\delta x} = \frac{f_i^+(x)}{f_i^+(x) + f_i^-(x)} \tag{25}$$

$$P(B|\text{sim}(G_i, G_j) = x) = \frac{f_i^-(x)\delta x}{f_i^+(x)\delta x + f_i^-(x)\delta x} = \frac{f_i^-(x)}{f_i^+(x) + f_i^-(x)}. \tag{26}$$

The derivative of $P(A|\text{sim}(x_i, x_j) = x)$ with respect to the similarity score x is:

$$P(A|\text{sim}(x_i, x_j) = x)' = \frac{f_i^{+'}(x)f_i^-(x) - f_i^{-'}(x)f_i^+(x)}{(f_i^+(x) + f_i^-(x))^2}. \tag{27}$$

We wish to find the monotonic non-decreasing interval of $P(A|\text{sim}(x_i, x_j) = x)$ w.r.t. the similarity score x. The existence of such interval indicates that it is reasonable to give graphs instances that have higher similarity scores with the target instance larger probability to be selected as its positive instances.

The question can be further changed to finding the similarity score interval where $\frac{f_i^{+'}(x)}{f_i^+(x)} > \frac{f_i^{-'}(x)}{f_i^-(x)}$ holds. Such interval may be determined by the shape of those two probability density functions and their respective parameters. Let us consider a specific function cluster: the truncated re-normalized normal distribution density function cluster. We explain such functions by giving an example as follows. Consider the probability density function of a normal distribution: $g_0(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$, its truncated re-normalized density function over $[0, 1]$ is $g(x) = \frac{g_0(x)}{\int_0^1 g_0(x)\text{d}x}$. It is obvious that $\int_0^1 g(x)\text{d}x = 1$. Consider the situation where both $f_i^+(\cdot)$ and $f_i^-(\cdot)$ are such truncated re-normalized normal distribution probability density functions with parameters (μ_+, σ_+) for $f_i^+(\cdot)$ and (μ_-, σ_-) for $f_i^-(\cdot)$. It is naturally to assume that $\mu_- < \mu_+$ to meet with the properties proposed in Def. 2. Thus, we have:

$$\frac{f_i^{+'}(x)}{f_i^+(x)} = -\frac{x - \mu_+}{\sigma_+^2} \tag{28}$$

$$\frac{f_i^{-'}(x)}{f_i^-(x)} = -\frac{x - \mu_-}{\sigma_-^2}. \tag{29}$$

Since we have no prior knowledge of the relationship between σ_+ and σ_-, we discuss the existence of the non-decreasing similarity score interval w.r.t. the relationship between σ_+ and σ_- as follows:

- Case 1. If $\sigma_+ = \sigma_-$, we have $\frac{f_i^{+'}(x)}{f_i^+(x)} > \frac{f_i^{-'}(x)}{f_i^-(x)}$ for every $0 \leq x \leq 1$;
- Case 2. If $\sigma_+ < \sigma_-$, which indicates that the similarity score distribution over the ground-truth grpah instance set is more centralized, $\frac{f_i^{+'}(x)}{f_i^+(x)} > \frac{f_i^{-'}(x)}{f_i^-(x)}$ can be satisfied when $0 \leq x < \min\left(\frac{\sigma_-^2\mu_+ - \sigma_+^2\mu_-}{\sigma_-^2 - \sigma_+^2}, 1\right)$;
- Case 3. If $\sigma_+ > \sigma_-$, $\frac{f_i^{+'}(x)}{f_i^+(x)} > \frac{f_i^{-'}(x)}{f_i^-(x)}$ can be satisfied when $\max\left(\frac{\sigma_+^2\mu_- - \sigma_-^2\mu_+}{\sigma_+^2 - \sigma_-^2}, 0\right) < x \leq 1$.

The limitation here is that we restrict the shape of the similarity score possibility distribution density functions as well as the relationship between their parameters to arrive at the above conclusion. It is possible that similar conclusions can be arrived at when $f_i^+(\cdot)$ and $f_i^-(\cdot)$ are in other forms.

B.4 Sampling preference brought by the high-order graph sampling strategy

In this section, we want to discuss into the sampling preference brought by the high-order sampling process towards those graph instances that are high-order connected to the target graph instance,

including those graph instances that are both high-order connected and also lower-order connected to the target graph instance as well as those that are only high-order connected to the target graph instance.

The connectivity between two nodes in the graph is introduced as follows:

Definition 3 (Connectivity)**.** *Node n_i and node n_j is k-connected, if and only if there exists a loop-free path with length k between them. For such two nodes n_i and n_j, a node sequence $n_0(=n_i), n_1, ..., n_k(=n_j)$ can be found, where $n_p \neq n_q, \forall p \neq q, 0 \leq p \leq k, 0 \leq q \leq k$. k is a connectivity order between node n_i and node n_j.*

We only take the second-order sampling strategy as an example here for its simplicity and generality. It is easy to generalize to high-order sampling strategy, since the properties of the high-order random walk, the high-order sampling strategy used in this paper, have been researched and discussed thoroughly. The ratio for selecting each first-order neighbour $G_j \in \mathcal{G}_i^{\text{sim}_\tau+}$ of the graph instance G_i when using the first-order sampling strategy is:

$$P_i^+(x_j) = \frac{\text{sim}(G_i, G_j)}{\sum_{G_k \in \mathcal{G}_i^{\text{sim}_\tau+}} \text{sim}(G_i, G_k)}. \tag{30}$$

For each graph instance $G_i^- \in \mathcal{G} \setminus \mathcal{G}_i^{\text{sim}_\tau+}$, we have $P_i^+(x_i^-) = 0$.

When using second-order sampling strategy, the corresponding sampling ratio is proportional to:

$$\hat{P}_i^{2+}(x_j) = P_i^+(x_j) + \sum_{G_k \in \mathcal{G}_i^{\text{sim}+}} P_i^+(x_k) P_k^+(x_j), \tag{31}$$

for each $G_j \in \mathcal{G}$.

Obviously, second-order sampling strategy gives larger preference for G_i's first-order neighbours that are also 2-connected to G_i than its neighbours that are only 1-connected to G_i, compared with the first-order sampling strategy[11].

Experimental evidence. Such sampling preference to 2-connected first-order neighbours can be verified by simple experimental results from the following aspects: 1). Similarity scores between G_i's first-order neighbours that are connected to each other selected by the second-order sampling strategy should be higher than that resulted by the first-order sampling strategy. It is because that if second-order sampling strategy tends to select 2-connected first-order neighbours more than only 1-connected neighbours, it is more likely that the chosen first-order neighbours are also connected with each other when using the second-order sampling strategy. It can be verified by experimental results shown in Table 19, the sampled first-order neighbours are more likely to be connected to each other and also more similar with each other when using second-order sampling strategy than using first-order sampling strategy. 2). Sampled first-order neighbours tend to be more similar with the target graph instance when using second-order sampling strategy than using first-order sampling strategy. It is not a straightforward conclusion that can be drawn by analyzing the sampling preference of the second-order sampling strategy for different kinds of neighbours, but can be seen from the experimental results (see the column "Target Sim." in Table 19). Thus, the second-order sampling strategy may also tend to sample more similar first-order neighbours, which may be more likely to be ground-truth positive instances.

However, second-order sampling also leads to the possibility to sample neighbours that are only 2-connected to the target graph instance, whose sampling rates are proportional to:

$$\hat{P}_i^{2+}(x_j) = \sum_{G_k \in \mathcal{G}_i^{\text{sim}+}} P_i^+(x_k) P_k^+(x_j). \tag{32}$$

Since such instances that are only 2-connected to the target graph instances are more likely to be false-positive instances, high-order sampling strategy may increase the risk of sampling false-positive instances.

[11] Note that this does not mean that the sampling ratio for neighbours that are both 1-connected and 2-connected to G_i is larger than that for neighbours that are only 1-connected to G_i.

Table 19: Statistical results for similarity scores within sampled positive instances, similarity scores for sampled positive instances with the target graph instance and the ratio of sampled positive instances connected to each other. Results for first-order neighbourhood sampling strategy and second-order sampling strategy are presented in the table, where only sampled first-order neighbours are chosen for calculation. In the experiment, we uniformly randomly sample 1000 molecules from the pre-training dataset and calculate the mean value of those three values. Three independent experiments were performed with mean and standard deviation values reported. Values presented in the table have the format mean ± std. For abbreviations used, "Inter-pos. Sim." denotes the similarity scores within sampled positive instances, "Target Sim." denotes the similarity scores for sampled instances with the target graph instance.

	Inter-pos. Sim.	Target Sim.	Connected Ratio
First-order	0.4552 ± 0.0026	0.4766 ± 0.0035	0.6171 ± 0.0108
Second-order	0.4687 ± 0.0057	0.4831 ± 0.0022	0.6857 ± 0.0302

B.5 Sampling bias brought by the approximate similarity score function

In this section, we want to discuss the possible sampling bias brought by the approximate similarity score function. The sampling bias may exist in two aspects: 1). Graph instance G_i's ground-truth positive instances that have higher similarity scores with G_i will enjoy larger sampling preference. Moreover, G_i's ground-truth positive instances with relatively low similarity scores are failed to be selected as its positive instances. However, they should be sampled equally when using the ground-truth similarity score function. 2). It is possible that G_i's negative instances that have relatively high similarity scores could be selected as its positive instances.

If we denote $P_i^{\text{gt}+}(\cdot)$ as graph G_i's ground-truth positive sampling ratio function, then G_i's positive sampling bias when using the approximate similarity function from the ground-truth similarity function is defined as:

$$\text{bias}_i^\tau = \sum_{G_k \in \mathcal{G}} |P_i^+(x_k) - P_i^{\text{gt}+}(x_k)| \tag{33}$$

$$= \sum_{G_k \in \mathcal{G}_i^{\text{gt}+}} |P_i^+(x_k) - P_i^{\text{gt}+}(x_k)| + \sum_{G_k \in \mathcal{G}_i^{\text{gt}-}} |P_i^+(x_k) - P_i^{\text{gt}+}(x_k)| \tag{34}$$

$$= \text{gap}_i^\tau + \text{risk}_i^\tau. \tag{35}$$

gap_i^τ and risk_i^τ are the functions of $f_i^+(\cdot)$, $f_i^-(\cdot)$ and τ:

$$\text{gap}_i^\tau = |\mathcal{G}_i^{\text{gt}+}| \int_0^\tau \frac{f_i^+(x)}{|\mathcal{G}_i^{\text{gt}+}|} \mathrm{d}x + |\mathcal{G}_i^{\text{gt}+}| \int_\tau^1 \left| \frac{x}{\text{totsim}_\tau} - \frac{1}{|\mathcal{G}_i^{\text{gt}+}|} \right| f_i^+(x) \mathrm{d}x \tag{36}$$

$$\text{risk}_i^\tau = |\mathcal{G}_i^{\text{gt}-}| \frac{\int_\tau^1 x f_i^-(x) \mathrm{d}x}{\text{totsim}_\tau}, \tag{37}$$

where $\text{totsim}_\tau = |\mathcal{G}_i^{\text{gt}+}| \int_\tau^1 x f_i^+(x) \mathrm{d}x + |\mathcal{G}_i^{\text{gt}-}| \int_\tau^1 x f_i^-(x) \mathrm{d}x$ is the sum of the similarity scores over G_i's positive instance candidates. Assume that:

- Ground-truth positive instances would always not be explored thoroughly, which means that $P_i^+(x_k) = \frac{\text{sim}(x_i, x_k)}{\text{totsim}_\tau} > \frac{1}{|\mathcal{G}_i^{\text{gt}+}|}$ for each $G_k \in \mathcal{G}_i^{\text{gt}+} \cap \mathcal{G}_i^{\text{sim}_\tau +}$.

This assumption is reasonable, since

- $|\mathcal{G}_i^{\text{gt}+}| \gg |\mathcal{G}_i^{\text{sim}_\tau +}|$, considering that the pre-training dataset is always large and while $|\mathcal{G}_i^{\text{sim}_\tau +}|$ is relatively small to reduce the similarity score computation budget for efficiency.

To meet with the assumption, we can further introduce τ_3 for each graph instance G_i, where $\frac{\tau_3}{\text{totsim}_{\tau_3}} = \frac{1}{|\mathcal{G}_i^{\text{gt}+}|}$, and restrict the similarity threshold τ to $\tau_3 < \tau < 1$ for each G_i's τ_3.

Thus, gap_i^τ becomes:

$$\text{gap}_i^\tau = |\mathcal{G}_i^{\text{gt}+}| \int_0^\tau \frac{f_i^+(x)}{|\mathcal{G}_i^{\text{gt}+}|} \mathrm{d}x + |\mathcal{G}_i^{\text{gt}+}| \int_\tau^1 \left(\frac{x}{\text{totsim}_\tau} - \frac{1}{|\mathcal{G}_i^{\text{gt}+}|} \right) f_i^+(x) \mathrm{d}x \tag{38}$$

$$= \left\{ \int_0^\tau f_i^+(x) \mathrm{d}x - \int_\tau^1 f_i^+(x) \mathrm{d}x \right\} + \left\{ |\mathcal{G}_i^{\text{gt}+}| \int_\tau^1 \frac{x f_i^+(x)}{\text{totsim}_\tau} \mathrm{d}x \right\} \tag{39}$$

We denote such two items as $\text{gap}_i^{\tau,\text{gt}} = \int_0^\tau f_i^+(x)\mathrm{d}x - \int_\tau^1 f_i^+(x)\mathrm{d}x$ and $\text{gap}_i^{\tau,\text{sim}_\tau} = |\mathcal{G}_i^{\text{gt}+}| \cdot \int_\tau^1 \frac{x f_i^+(x)}{\text{totsim}_\tau}\mathrm{d}x$ respectively.

If we assume that $f_i^+(\cdot)$ and $f_i^-(\cdot)$ are truncated re-normalzied normal distribution functions and further restrict the relationship between their parameters, we can show that risk_i^τ increases as the hreshold τ increases, thus at the same time $\int_\tau^1 \frac{x f_i^+(x)}{\text{totsim}_\tau}\mathrm{d}x$ decreases as τ increases since $\frac{|\mathcal{G}_i^{\text{gt}+}| \int_\tau^1 x f_i^+(x)\mathrm{d}x}{\text{totsim}_\tau} + \frac{|\mathcal{G}_i^{\text{gt}-}| \int_\tau^1 x f_i^-(x)\mathrm{d}x}{\text{totsim}_\tau} = 1$, though may not that intuitive:

$$\text{risk}_i^\tau = \frac{|\mathcal{G}_i^{\text{gt}-}| \int_\tau^1 x f_i^-(x)\mathrm{d}x}{|\mathcal{G}_i^{\text{gt}-}| \int_\tau^1 x f_i^-(x)\mathrm{d}x + |\mathcal{G}_i^{\text{gt}+}| \int_\tau^1 x f_i^+(x)\mathrm{d}x} \tag{40}$$

$$\frac{\partial \text{risk}_i^\tau}{\partial \tau} = \frac{|\mathcal{G}_i^{\text{gt}+}||\mathcal{G}_i^{\text{gt}-}|\tau(f_i^+(\tau)\int_\tau^1 x f_i^-(x)\mathrm{d}x - f_i^-(\tau)\int_\tau^1 x f_i^+(x)\mathrm{d}x)}{(|\mathcal{G}_i^{\text{gt}-}| \int_\tau^1 x f_i^-(x)\mathrm{d}x + |\mathcal{G}_i^{\text{gt}+}| \int_\tau^1 x f_i^+(x)\mathrm{d}x)^2}, \tag{41}$$

where the sign of $\frac{\partial \text{risk}_i^\tau}{\partial \tau}$ is determined by the sign of $f_i^+(\tau)\int_\tau^1 x f_i^-(x)\mathrm{d}x - f_i^-(\tau)\int_\tau^1 x f_i^+(x)\mathrm{d}x$. For $f_i^+(\cdot)$'s parameters (μ_+, σ_+) and $f_i^-(\cdot)$'s parameters (μ_-, σ_-), we assume that $\sigma_+ = \sigma_- = \sigma, \mu_+ > \mu_-$. Thus,

$$\frac{\int_\tau^1 x f_i^-(x)\mathrm{d}x}{f_i^-(\tau)} = \frac{\int_\tau^1 x \exp\left(\frac{(x-\mu_-)^2}{2\sigma^2}\right)\mathrm{d}x}{\exp\left(\frac{(\tau-\mu_-)^2}{2\sigma^2}\right)} = \int_\tau^1 x \exp\left(\frac{x^2 - 2\mu_-(x-\tau) - \tau^2}{2\sigma^2}\right)\mathrm{d}x \tag{42}$$

$$\frac{\int_\tau^1 x f_i^+(x)\mathrm{d}x}{f_i^+(\tau)} = \frac{\int_\tau^1 x \exp\left(\frac{(x-\mu_+)^2}{2\sigma^2}\right)\mathrm{d}x}{\exp\left(\frac{(\tau-\mu_+)^2}{2\sigma^2}\right)} = \int_\tau^1 x \exp\left(\frac{x^2 - 2\mu_+(x-\tau) - \tau^2}{2\sigma^2}\right)\mathrm{d}x. \tag{43}$$

We have,

$$\int_\tau^1 x \exp\left(\frac{x^2 - 2\mu_-(x-\tau) - \tau^2}{2\sigma^2}\right)\mathrm{d}x > \int_\tau^1 x \exp\left(\frac{x^2 - 2\mu_+(x-\tau) - \tau^2}{2\sigma^2}\right)\mathrm{d}x \tag{44}$$

since $x - \tau \geq 0, \forall \tau \leq x \leq 1$ and $\mu_- < \mu_+$.

Thus, $\text{gap}_i^{\tau,\text{gt}}$ and risk_i^τ decrease as τ decreases, while $\text{gap}_i^{\tau,\text{sim}_\tau}$ increases as τ decreases. Though it is hard to detect the monotonicity of gap_i^τ, we can get a sense that there exists a balance between minimizing gap_τ and risk_τ.

A limitation that must be pointed out about the above conclusion, including the monotonicity of each part in gap_i^τ and risk_i^τ, is arrived by assuming the similarity possibility density functions $f_i^+(\cdot)$ and $f_i^-(\cdot)$ are chosen from a certain function cluster and further restricting the relationship between their parameters. Thus, the above conclusion only aims at giving a glimpse into the potential balance existing in the sampling bias brought by the approximate similarity function over the ground-truth positive instances and negative instances.

Though it seems that the high-order sampling strategy is not taken into consideration in the above discussion, it can be naturally integrated in since the positive similarity threshold τ is lowered in the high-order sampling process. Moreover, the abstracted similarity threshold τ implies that we can probably reach a good balance point, which may be determined by the specific application scenario, by tuning the similarity score threshold τ directly or changing the positive instance sampling strategy.

C Broader Impact

In this paper, we have developed a sampling based graph positive instance selection strategy (HGC) that can be used in the graph contrastive learning process. Compared with previous approaches, where positive instances for the target graph instance are obtained by performing data augmentation skills on the target graph instance, our sampling based process can ultimately get positive graph instances of better quality keeping enough similarity with the target graph instance and also within different positive graph instances. Such a sampling process can also ensure the necessary domain specific information preserved in the resulting positive instances. We also propose an improvement

on a widely used node-level pre-training strategy (AdaM), leading to masked nodes distributed more evently on the graph. Moreover, we discover the potential possibility of the pre-trained GNN models to perform cross-domain transferring.

It seems that there are no explicit relationship between our graph-level similarity based positive instance sampling strategy and our improvement for the widely used attribute masking pre-training strategy. However, we want to point out that their design principles point towards a higher methodology design philosophy. That is, introducing prior knowledge into methods that are originally random ones can help us get better results. Specifically, we introduce the approximate pair-wise similarity information which can help us sample positive instances of better quality. By comparison, previous methods always tend to use data augmentation methods to construct positive graph instances from the target instance. Such strategies always introduce random factors to perturb the structure and attribute information in the graph. Those random factors may destruct necessary information that should be kept in the positive instances. Admittedly, it is also possible to design data augmentation strategies that are aware of such information to preserve them in the resulting graph instances. However, it will turn out to be a complex strategy with many restrictions on the augmentation process. The advantage of our HGC is then obvious – it keeps such necessary information in the positive samples by sampling from existing graphs with some deterministic factors fused into the sampling process automatically (transition probability from one node in the graph to another node is calculated by their pair-wise similarity score). Thus, it is a more effective, efficient and elegant solution for such a crucial problem existing in contrastive learning for graph data.

Our adaptive masking strategy tries to select nodes based on their perturbation scores, which ultimately leads to nodes selected distributed more evenly in the graph. It is inspired by Kmeans++ [2]. Though we still aim at distributing nodes evenly in the graph, we approximately solve such a problem by adding some deterministic factors (the sampling rates for remaining nodes are calculated based on their perturbation scores) in the node selecting process, different from previous methods which just selecting nodes according to a uniform distribution.

The broader impact of our research can be summarized below:

- **For machine learning community:** This work demonstrates the importance of designing machine learning strategies by thinking deeply into essential things that are most important to solve the problem (e.g., how to ensure the enough similarity between positive instances and target instances in our positive instance selection problem). The sampling based positive instance selection process may probably inspire more novel graph instance pre-training strategies.
  Moreover, we point out a potential new developing direction for pre-training on graph data. That is, how can we obtain powerful universally transferrable pre-trained GNN models that can transfer across different kinds of graphs? It is an interesting and also a valuable quation that deserves further discussion.
- **For the drug discovery community:** Researchers from the drug discovery community can benefit from this work. It is because that the starting point of the design of HGC is the wish to apply contrastive learning strategy on molecular graphs effectively, since previous approaches may impede the development of contrastive learning for pre-training on molecular graphs, which are kind of special graphs in the real-world. Thus, the contrastive learning using HGC for positive instance sampling can help with developing GNN pre-training strategies. We hope that HGC can help with boosting the performance of various drug discovery applications, such as molecular property prediction and virtual screening.