A. Proof of Theorem 1

Proof : Since $\boldsymbol{U}$ is invertible, we can alternatively optimize $\boldsymbol{P} = \boldsymbol{U}^T\boldsymbol{W}$ instead of $\boldsymbol{W}$ in the contrastive loss. Then the optimization in Eq. (4) can be reformalized as

$$\min_{\boldsymbol{P}} \text{tr}(\boldsymbol{M}\boldsymbol{M}^T), \quad \text{subject to} \quad \boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{I}, \tag{9}$$

where $\boldsymbol{M} = \boldsymbol{U}(\Lambda^L - \Lambda^{L'})\boldsymbol{P}$. To solve the optimization problem, we first relax the condition to $\boldsymbol{P}_i^T\boldsymbol{P}_i = 1$ for $i = 1, 2 \ldots d_O$, where $\boldsymbol{P}_i$ is the i-th column of $\boldsymbol{P}$. Then the Lagrangian function is

$$\begin{aligned}&\mathcal{L}(\boldsymbol{P}, \beta)\\&= \text{tr}(\boldsymbol{U}(\Lambda^L - \Lambda^{L'})\boldsymbol{P}\boldsymbol{P}^T(\Lambda^L - \Lambda^{L'})\boldsymbol{U}^T) - \sum_{i=1}^{d_O}\beta_i(\boldsymbol{P}_i^T\boldsymbol{P}_i - 1)\\&= \text{tr}((\Lambda^L - \Lambda^{L'})^2\boldsymbol{P}\boldsymbol{P}^T) - \sum_{i=1}^{d_O}\beta_i(\boldsymbol{P}_i^T\boldsymbol{P}_i - 1),\end{aligned} \tag{10}$$

where $\beta = (\beta_1, \beta_2 \ldots \beta_{d_O})$ is the Lagrangian multiplier.

With the Karush−Kuhn−Tucker (KKT) conditions, we have

$$\begin{aligned}\frac{\partial\mathcal{L}(\boldsymbol{P}, \beta)}{\partial\boldsymbol{P}} &= 0\\2(\Lambda^L - \Lambda^{L'})^2\boldsymbol{P} - 2\boldsymbol{P}\text{diag}(\beta) &= 0\\(\Lambda^L - \Lambda^{L'})^2\boldsymbol{P} &= \boldsymbol{P}\text{diag}(\beta).\end{aligned} \tag{11}$$

Therefore, the columns of $\boldsymbol{P}$ are the eigenvectors of matrix $(\Lambda^L - \Lambda^{L'})^2$. Note that the original condition $\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{I}$ requires that the columns of $\boldsymbol{P}$ are orthogonal. Hence the optimal $\boldsymbol{P}$ is the eigenvectors of $(\Lambda^L - \Lambda^{L'})^2$ with minimum eigenvalues:

$$\boldsymbol{P}^* = \boldsymbol{I}[k_1, k_2, \ldots k_{d_O}], \tag{12}$$

where $1 \le k_1 < k_2 < \ldots k_{d_O} \le |\mathcal{V}|$ are the best ks that minimize $(\lambda_k^L - \lambda_k^{L'})^2$, and $\boldsymbol{I}[k_1, k_2, \ldots k_{d_O}]$ denotes the corresponding columns of the identity matrix $\boldsymbol{I}$. QED.

B. Additional Experiments

Graph Classification Experiments

Table 4: Datasets statistics for graph classification.

Dataset	#Graphs	Avg. #Nodes	Avg. #Edges	#Classes
NCI1	4,110	29.87	32.3	2
MUTAG	188	17.93	19.79	2
COLLAB	5,000	74.5	2457.78	3
IMDB-BINARY	1,000	13	65.94	2

Though our strategies are mostly derived from node-level representation learning, we conduct experiments to further validate our idea of model augmentation on 4 graph classification benchmarks: NCI1, MUTAG, COLLAB and IMDB-BINARY in TUDatasets (Morris et al. 2020). The statistics of datasets can be found in Table 4. Specifically, we apply the three strategies to three state-of-the-art GCL models for graph-level representation learning, including GraphCL (You et al. 2020), AD-GCL (Suresh et al. 2021) and SimGRACE (Xia et al. 2022). Then we have their model augmented versions as GraphCL+MA, AD-GCL+MA and SimGRACE+MA, and compare the classification performance with their original models. We fix the range of K as $[0, 2]$ and the range of K' as $[0, 4]$. For AD-GCL+MA, we remove the component of adaptive augmentations in AD-GCL, and only employ the backbone with feature and edge dropping. We use GIN (Xu et al. 2018) as the encoder for all methods and all other experimental settings are the same as original models (You et al. 2020; Suresh et al. 2021; Xia et al. 2022). We report the results in Fig. 3. As we can see in the results, our model augmentation mechanism can be successfully applied to the SOTA GCL models for graph-level modeling, and achieve better performance on downstream graph classification tasks.

Node Clustering

We adopt the K-means algorithm for node clustering, and use the implementation by kmeans-pytorch. We set the number of cluster centers of k-means algorithm as the number of classes in classification task and calculate the normalized mutual information (NMI) to evaluate the results of clustering on six datasets. Clustering results are the median over 20 randomly initialized runs and the results are report in Table 5 and MA-GCL can achieve SOTA performance on 5 out of 6 datasets.

Motivation Verification Experiments

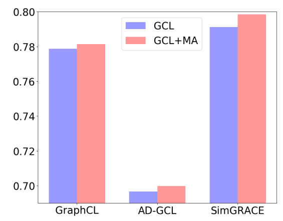
We present motivation verification experiments on Cora, PubMed Amazon-C and Coauthor-CS in Fig. 4. All the settings are the same as in Section . The patterns on Cora, PubMed, Amazon-C and Coauthor-CS are also consistent with those in previous section.

Hyper-parameter Experiments

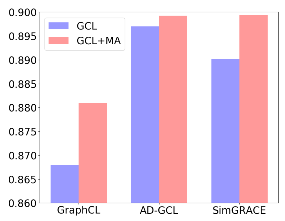
Effect of Random Ranges We explore the influence of the ranges of K and K' on node classification. Here we try different range combinations. Specifically, we fix the low and low' as 0, and report the corresponding performance under different combinations of $high$ and $high'$ on Cora, CiteSeer and PubMed (random splits). Here the performance is the average accuracy over 5 runs in different random seeds. The experimental results are shown in Fig. 5. We can find that (1) By simply setting the range of K as $[0, 2]$ and the range of K' as $[0, 8]$, MA-GCL can achieve SOTA performance on all the three benchmarks. The results are even better than we reported in the main document. According to our theory of the asymmetric strategy, a wider gap between K and K' can help better alleviate high-frequency noises. Also note that negative samples will adopt the view encoder corresponding to K instead of K'. Hence for the range pair of K and K', $[2, 8]$ is much better than $[8, 2]$ due to the over-smoothing issue. (2) MA-GCL is not very sensitive to the ranges of K and K'. The difference between the best accuracy and the worst one is less than one percent on three benchmarks.

Effect of Hidden Size of Encoder and Projector We explore the effects of the hidden embedding size of graph encoder and projector on node classification tasks, and the experiments are conducted on three citation networks of Cora, CiteSeer and PubMed. Specifically, we fix other hyper-parameters and change the hidden size of encoder within the value of $\{64, 128, 256, 512\}$ and report the trend of performance in Fig .6(a). Similarly, we also conduct the same experiments with the hidden size of projector, and report the results in Fig .6(b). All of other settings are same as mentioned in . As we can see in Fig .6, as the complexity of the model increases, the performance of the node classification increases smoothly, and our model can still achieve impressive performance with simple encoder and projector with few parameters.

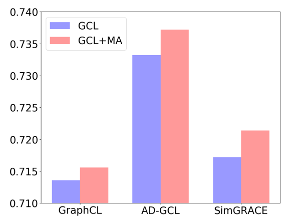
Effect of Graph Filters We conduct experiments to explore the effects of graph filter $\boldsymbol{F}$ in MA-GCL. Note that we use fixed $\boldsymbol{F} =$



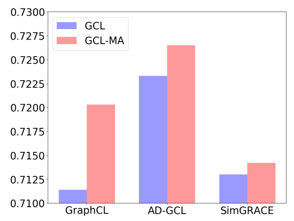
(a) NCI1



(b) MUTAG



(c) COLLAB

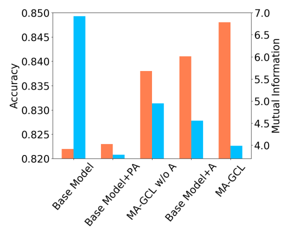


(d) IMDB-B

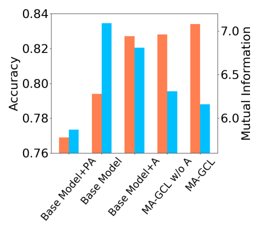
Figure 3: Experimental results of graph classification accuracy on 4 datasets. Every pair of columns represent an original GCL model and its variant with our model augmentation mechanism, respectively.

Table 5: Performance of Node Clustering

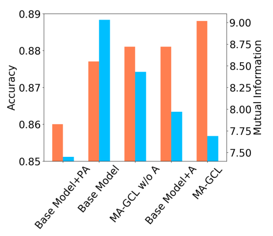
NMI	Cora	Citeseer	PubMed	Amazon-P	Amazon-C	Coauthor-CS
raw feature	0.144	0.199	0.178	0.261	0.194	0.563
Base Model	0.471	0.429	0.221	0.535	0.474	0.601
GCA	0.474	0.441	0.224	0.587	0.499	0.614
S2GC	0.502	0.449	0.257	0.578	0.513	0.679
COLES	0.470	0.415	0.262	0.546	0.497	0.660
ARIEL	0.487	0.472	0.276	0.569	0.525	0.653
CCA-SSG	**0.517**	0.476	0.254	0.591	0.517	0.677
MA-GCL	0.509	**0.494**	**0.297**	**0.602**	**0.533**	**0.696**



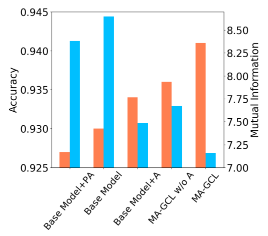
(a) Cora



(b) PubMed

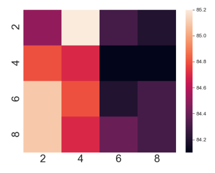


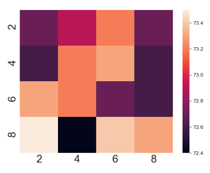
(c) Amazon-Computers



(d) Coauthor-CS

Figure 4: Experimental results of motivation verification.


(a) Cora


(b) CiteSeer

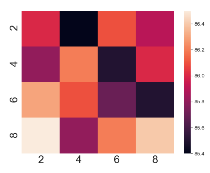
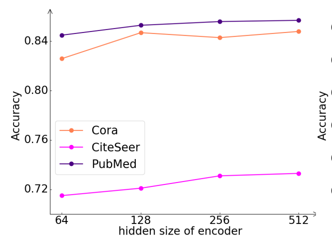
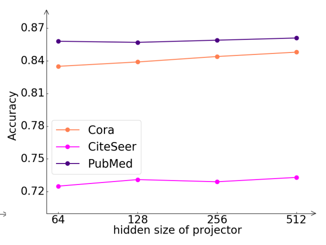

(c) PubMed

Figure 5: Visualizations of the accuracies under different range combinations of $K \in [0, high]$ and $K' \in [0, high']$. Each row represents a specific setting for $high'$ and each column represents a specific setting for $high$.



(a) hidden size



(b) projector size

Figure 6: Visualization of the trend of accuracy with respect to hidden size of encoder and projector. Lines with different colors represent the results on different datasets.

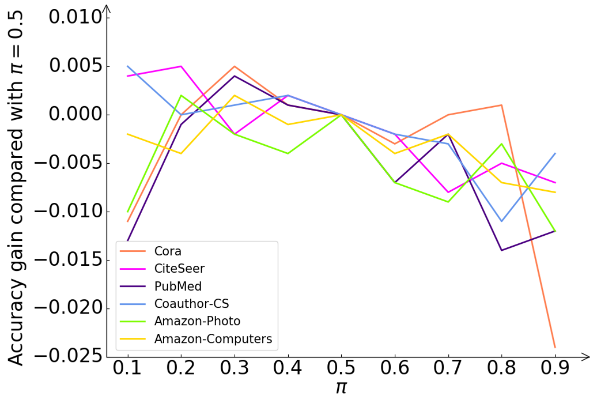


Figure 7: Comparisons among graph filters with different π. Lines with different colors represent the results on different datasets. For convenience, we subtract the accuracy of $\pi = 0.5$ for each line.

$(1-\pi)\boldsymbol{I} + \pi \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ by setting $\pi = 0.5$ in previous sections. In this section, we try different graph filters by choosing $\pi \in (0, 1)$, and report the results on the 6 benchmarks. We randomly split the datasets, where 10%, 10%, and the rest 80% of nodes are selected for training, validation and test sets. We report the average accuracy over 5 runs in different random seeds. Other hyper-parameters are the same as reported in Table 7 and 8. Experimental results are shown in Fig. 7. We find that the optimal π of every dataset is less than 0.5. Recall that the eigenvalues of $\boldsymbol{F}$ approximately fall in range $[1-\frac{3}{2}\pi, 1]$, and the function $\min_\lambda(\lambda^L - \lambda^{L'})^2$ prefers $\lambda \to 0$ or $\lambda \to 1$. Hence a smaller π can help GCL preserve the information corresponding to $\lambda \to 1$, and filter out high-frequency noise (*i.e.*, the graph signals corresponding to $\lambda \to 0$). This observation can also validate our theory of the asymmetric strategy.

Time and Space Efficiency

Here we will show whether the number of propagation layers L will affect the efficiency. In fact, the space complexity will not change as L increases, and the time complexity is only sublinear to L. Taking the Coauthor-CS dataset as an example, the memory usages of $L = 2$ and $L = 10$ are both 7.5GB. And the training times of a single epoch are 0.053s and 0.113s, respectively. Thus MA-GCL is very efficient.

C. More Detailed Experimental Settings

Hyper-parameter Settings

We provide all the detailed hyper-parameters settings on the six datasets in Table 7 and 8, where EDR and FDR are the edge dropping rate and feature dropping rate, respectively. Note that for the linear classifier in evaluation, we use the fixed learning rate of 0.01 and weight decay of 0 for fair comparisons. For most hyper-parameters, we follow the settings of GCA (Zhu et al. 2021) and all hyper-parameters are heuristically searched in the following space:

- The number of training epoches: $\{300, 400, 500, 800, 900, 1,000, 1,500, 2,000\}$
- Range of K: $[low, high]$ for $low, high \in \{0, 1, 2, 3, 4, 5\}$
- Hidden size of h operator: $\{64, 128, 256, 512\}$
- Hidden size of projector: $\{64, 128, 256, 512\}$
- Learning rate: $\{0.001, 0.002, 0.005, 1e-4, 2e-4, 5e-4, 1e-5, 2e-5, 5e-5\}$
- Weight decay: $\{1e-4, 1e-5, 1e-6, 0\}$
- Active function: $\{ReLU, PReLU, RReLU\}$
- Edge dropping rate and feature dropping rate: $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$

Datasets

We evaluate our models on six node classification benchmarks include Cora, CiteSeer, PubMed, Coauthor-CS, Amazon-Computer and Amazon-Photo. The statistics of datasets is shown in Table 6.

Cora, CiteSeer, PubMed are three commonly used node classification datasets, each of which contains one citation network, where nodes mean papers and edges represent citation relationships.

Coauthor-CS is a widely used node classification benchmarks based on the Microsoft Academic Graph from the KDD Cup 2016 challenge (Sinha et al. 2015). The vertices represent authors and edges mean the co-author relationship between authors. The ground-truth labels of authors indicate most active fields of study for authors, and the keywords of the author's papers are extracted as the features of the vertices.

Amazon-Photo, Amazon-Computers are node classification datasets based on the Amazon co-purchase graph. The nodes mean goods and edges represent that two items are often purchased together. The node attributes are bag-of-words extracted from product reviews, and the product category determines the labels of nodes.

We use the pre-processed version provided by PyTorch-Geometric (Fey and Lenssen 2019) for all datasets.

Classifier

For all the experiments, we follow the linear evaluation scheme as introduced in DGI (Velickovic et al. 2019). Specifically, models is firstly trained with an unsupervised manner. After that, the learned representations are used to train and test a simple l_2-regularized logistic regression classifier with fixed learning rate and weight decay of 0.01 and 0 respectively.

Environment

We implement our method with PyTorch and PyTorch-Geometric (Fey and Lenssen 2019), we adopt the Adam optimizer (Kingma and Ba 2015) for parameter learning. All experiments are conducted on a NVIDIA GeForce RTX 3090 GPU with 24 GB memory.

Table 6: Statistics of datasets.

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2,708	10,556	133	7
CiteSeer	3,327	9,228	3,703	6
PubMed	19,717	88,651	500	3
Coauthor-CS	18,333	327,576	6,805	15
Amazon-P	7,650	287,326	745	8
Amazon-C	13,752	574,418	767	10

Table 7: Settings of major hyper-parameters

Dataset	epoch	K Range	K' Range	hidden_size	proj_size
Cora	500	[0,4]	[1,4]	512	512
CiteSeer	400	[2,4]	[1,3]	512	512
PubMed	900	[0,3]	[0,3]	512	128
Coauthor-CS	1,000	[0,3]	[1,3]	256	256
Amazon-P	1,500	[3,5]	[1,3]	256	256
Amazon-C	2,000	[1,4]	[0,3]	256	256

Table 8: Settings of minor hyper-parameters

Dataset	lr	wd	activation	EDR_1	EDR_2	FDR_1	FDR_2
Cora	2e-4	1e-06	ReLU	0.3	0.3	0.3	0.3
CiteSeer	1e-5	1e-06	ReLU	0.3	0.2	0.3	0.2
PubMed	0.002	1e-05	ReLU	0.5	0.4	0.3	0.5
Coauthor-CS	5e-4	1e-05	RReLU	0.3	0.2	0.3	0.4
Amazon-P	0.001	0	PReLU	0.3	0.4	0.2	0.3
Amazon-C	0.001	1e-05	RReLU	0.7	0.2	0.2	0.2