

A DETAILS OF DATASETS

We conduct our experiments on 21 different graph real-world datasets for graph classification tasks. In this section, we provide detailed descriptions of the datasets used in this paper. Specifically, for the *supervised* learning setting, we include a total of eight datasets from the TUDatasets benchmark (Morris et al., 2020) (i.e., PROTEINS, NCI1, IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, REDDIT-MULTI-5K, and REDDIT-MULTI-12K) and the OGB benchmark (Hu et al., 2020a) (i.e., ogbg-molhiv). For the *semi-supervised* learning setting, we include seven different datasets from the TUDatasets benchmark (Morris et al., 2020) (i.e., PROTEINS, NCI1, DD, COLLAB, GITHUB, REDDIT-BINARY, and REDDIT-MULTI-5K). For the *unsupervised* learning setting, we include seven different datasets from the TUDatasets benchmark (Morris et al., 2020) (i.e., PROTEINS, NCI1, DD, MUTAG, IMDB-BINARY, REDDIT-BINARY, and REDDIT-MULTI-5K). The detailed of these datasets can be found in Table 5.

Table 5: Statistical characteristics of the datasets in three learning settings. Supe.: Supervised learning. Semi.: Semi-supervised learning. Unsu.: Unsupervised learning.

Dataset	Category	# Graphs	# Avg edges	# Classes	Task		
					Supe.	Semi.	Unsu.
IMDB-BINARY	Social Networks	1,000	96.53	2	✓		✓
IMDB-MULTI	Social Networks	1,500	65.94	3	✓		
REDDIT-BINARY	Social Networks	2,000	497.75	2	✓	✓	✓
REDDIT-MULTI-5K	Social Networks	4,999	594.87	5	✓	✓	✓
REDDIT-MULTI-12K	Social Networks	11,929	456.89	11	✓		
COLLAB	Social Networks	5,000	2457.78	3		✓	
GITHUB	Social Networks	12,725	234.64	2		✓	
DD	Biochemical Molecules	1,178	715.66	2		✓	✓
MUTAG	Biochemical Molecules	188	19.79	2			✓
PROTEINS	Biochemical Molecules	1,113	72.82	2	✓	✓	✓
NCI1	Biochemical Molecules	4,110	32.30	2	✓	✓	✓
ogbg-molhiv	Biochemical Molecules	41,127	27.50	2	✓		

For the *transfer* learning setting, we pre-train on ZINC-2M chemical molecule dataset (Sterling & Irwin, 2015; Gómez-Bombarelli et al., 2018), and fine-tune on eight different datasets, namely BBBP, Tox21, ToxCast, SIDER, ClinTox, MUV, HIV, and BACE. The detailed of these datasets can be found in Table 6.

Table 6: Statistical characteristics of the datasets used in the transfer learning setting.

Dataset	Strategy	# Molecules	# Binary tasks
ZINC-2M	Pre-training	2,000,000	-
BBBP	Fine-tuning	2,039	1
Tox21	Fine-tuning	7,831	12
ToxCast	Fine-tuning	8,576	617
SIDER	Fine-tuning	1,427	27
ClinTox	Fine-tuning	1,477	2
MUV	Fine-tuning	93,087	17
HIV	Fine-tuning	41,127	1
BACE	Fine-tuning	1,513	1

B DETAILS OF BASELINES

Supervised learning. For experiments in the supervised setting, we select the following baseline:

- DropEdge (Rong et al., 2019) selectively drops a portion of edges from the input graphs.
- DropNode (Feng et al., 2020) omits a specific ratio of nodes from the provided graphs.
- Subgraph (You et al., 2020) procures subgraphs from the main graphs using a random walk sampling technique.
- M-Mixup (Verma et al., 2019) blends graph-level representations through linear interpolation.
- SubMix (Yoo et al., 2022) combines random subgraphs from paired input graphs.

- G-Mixup (Han et al., 2022) employs a class-focused graph mixup strategy by amalgamating graphons across various classes.
- S-Mixup (Ling et al., 2023) adopts a mixup approach for graph classification, emphasizing soft alignments.

Semi-supervised learning. For experiments in the semi-supervised setting, we select the following baseline methods:

- GAE (Kipf & Welling, 2016b) is a non-probabilistic graph auto-encoder model, which is a variant of the VGAE (variational graph autoencoder).
- Informax (Veličković et al., 2018) trains a node encoder to optimize the mutual information between individual node representations and a comprehensive global graph representation.
- GraphCL (You et al., 2020) conducts an in-depth exploration of graph structure augmentations, including random edge removal, node dropping, and subgraph sampling.

Unsupervised learning. For experiments in the unsupervised setting, we select the following baseline methods:

- GL (graphlet kernel) (Pržulj, 2007) measures the similarity between graphs by counting the occurrences of small subgraphs, known as graphlets, within them. It captures local topological patterns, thus providing a comprehensive view of the graph structure.
- WL (Weisfeiler-Lehman sub-tree kernel) (Shervashidze et al., 2011) captures the similarity between graphs by comparing subtrees of increasing heights. It effectively distinguishes non-isomorphic graphs and is often employed for graph classification tasks.
- DGK (deep graph kernel) (Yanardag & Vishwanathan, 2015) combines the strengths of both graph kernels and deep learning. It leverages convolutional neural networks to learn hierarchical representations of graphs, enabling the kernel to capture complex patterns and structures within the data for a more refined similarity measure.
- node2vec (Grover & Leskovec, 2016) captures low-dimensional embeddings of graph nodes by leveraging random walks originating from target nodes.
- sub2vec (Adhikari et al., 2018) seeks to capture feature representations of arbitrary subgraphs, addressing the limitations of node-centric embeddings.
- graph2vec (Narayanan et al., 2017) is a neural embedding framework designed to learn data-driven distributed representations of entire graphs.
- InfoGraph (Sun et al., 2019) is designed to maximize the mutual information between complete graph representations and various substructures, such as nodes, edges, and triangles.
- GraphCL (see above section).
- MVGRL (Hassani & Khasahmadi, 2020) establishes a link between the local Laplacian matrix and a broader diffusion matrix by leveraging mutual information. This approach yields representations at both the node and graph levels, catering to distinct prediction tasks.
- AD-GCL (Suresh et al., 2021) emphasize preventing the capture of redundant information during training. They achieve this by optimizing adversarial graph augmentation strategies in GCL and introducing a trainable non-i.i.d. edge-dropping graph augmentation.
- JOAO (You et al., 2021) utilize a bi-level optimization framework to sift through optimal strategies, exploring multiple augmentation types like uniform edge or node dropping and subgraph sampling.
- GCL-SPAN (Lin et al., 2022) introduces a spectral augmentation approach, which directs topology augmentations to maximize spectral shifts.

Transfer learning. For experiments in the transfer setting, we select the following baseline methods:

- Informax (see above section).

- EdgePred (Hamilton et al., 2017) employs an inductive approach that utilizes node features, such as text attributes, to produce node embeddings by aggregating features from a node’s local neighborhood, rather than training distinct embeddings for each node.
- AttrMasking (Attribute Masking) (Hu et al., 2020b) is a pre-training method for GNNs that harnesses domain knowledge by discerning patterns in node or edge attributes across graph structures.
- ContextPred (Context Prediction) (Hu et al., 2020b) is designed for pre-training GNNs that simultaneously learn local node-level and global graph-level representations via subgraphs to predict their surrounding graph structures.
- GraphCL (see above section).

C DETAILS OF EXPERIMENTS SETTINGS

We conduct our experiments with PyTorch 1.13.1 on a server with NVIDIA RTX A5000 and CUDA 12.2. For each experiment, we run 10 times. We detail the settings of our experiments in this paper as follows.

Speed up implementation. Our augmentation method involves matrix eigenvalue decomposition, which is highly CPU-intensive. During implementation, we observed that when there is insufficient CPU, parallelly executing numerous matrix eigenvalue decomposition can lead to CPU resource deadlock. To address this issue, we established an additional set of CPU locks to manage CPU scheduling. Let N_{cpu} represent the number of CPUs available for each matrix eigenvalue decomposition task, and $N_{parallel}$ denote the number of decomposition tasks that can be executed simultaneously. We set $N_{cpu} \times N_{parallel}$ to be less than the total CPU number of the server. During task execution, we created a list of CPU locks, with each lock corresponding to N_{cpu} available CPUs. There is no overlap between the CPUs corresponding to each lock. Before a matrix decomposition task is executed, it must first request a CPU lock. Once the lock is acquired, the task can only be executed on the designated CPUs. After completion, the task releases the CPU lock. If there are no free locks in the current CPU lock list, the matrix decomposition task must wait. By employing this approach, we effectively isolated parallel matrix decomposition tasks.

Empirical studies. We first detail the processes and settings of the empirical studies below.

- **Experiment of Figure 1.** For DropEdge, 20% edges are randomly dropped. For DP-Noise, we use a standard deviation of 7, an augmentation probability of 0.5, and an augmentation frequency ratio of 0.5. For DP-Mask, the augmentation probability is set to 0.3, with an augmentation frequency ratio of 0.4. Detailed variations in edge numbers and properties between the original and augmented graphs can be found in Table 7.
- **Experiment of Figure 3.** In Figures 3a and 3b, labels in REDDIT-MULTI-12K for Class A and Class B are 1 and 10, respectively. The added edges in 3c are $3 \leftrightarrow 5$, $3 \leftrightarrow 7$, $1 \leftrightarrow 6$, $2 \leftrightarrow 6$, $0 \leftrightarrow 2$, $1 \leftrightarrow 3$, $1 \leftrightarrow 4$, $2 \leftrightarrow 4$, $4 \leftrightarrow 6$, and $5 \leftrightarrow 7$, respectively. The dropped edges in 3d are $0 \leftrightarrow 4$, $3 \leftrightarrow 4$, $4 \leftrightarrow 5$, $4 \leftrightarrow 7$, $0 \leftrightarrow 1$, $2 \leftrightarrow 3$, $0 \leftrightarrow 3$, $5 \leftrightarrow 6$, $6 \leftrightarrow 7$, and $1 \leftrightarrow 2$, respectively. The change of spectrum ΔL_2 is the L_2 distance between the spectrum of \mathcal{G} and augmented graph \mathcal{G}' , denoted as $\Delta L_2 = \sqrt{\sum_i (\lambda_i(\mathcal{G}) - \lambda_i(\hat{\mathcal{G}}))^2}$, where $\lambda(\mathcal{G})$ represent the spectrum of \mathcal{G} and $\lambda(\hat{\mathcal{G}})$ is the spectrum of $\hat{\mathcal{G}}$.
- **Experiment of Figure 5.** For both Figure 5a and 5b, we employ GIN as the backbone model and conduct experiments over five runs. In Figure 5b, while testing one parameter, we draw the other parameters from their respective search spaces: $\sigma \in [0.1, 0.5, 1.0, 2.0]$, $r_f \in [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8]$, and $r_a \in [0, 0.2, 0.4, 0.6, 0.8, 1]$. In addition, to control the noise adding to low- and high-frequency eigenvalues are in the same scale, the augmented i -th eigenvalue is calculated as $\lambda_i = \max(0, 1 + \epsilon) \times \lambda_i$, where $\epsilon \sim \mathcal{N}(0, \sigma)$.

Supervised learning. Following prior works (Han et al., 2022; Ling et al., 2023), we utilize two GNN models, namely GCN (Kipf & Welling, 2017) and GIN (Xu et al., 2018). Details of these GNNs are provided below.

- **GCN.** For the TUDatasets benchmark, the backbone model has four GCN layers, utilizes a global mean pooling readout function, has a hidden size of 32, and uses the ReLU activation function. For the ogbg-molhiv dataset, the model consists of five GCN layers, a hidden size of 300, the ReLU activation function, and a global mean pooling readout function.

Table 7: Details of alterations of the number of edges and properties of graphs in Figure 1.

Graph	Edge Alterations		Properties				
	# Dropped	# Added	Connectivity	Diameter	Radius	# Periphery	ASPL
Original	-	-	TRUE	2	1	11	1.42
DropEdge	7	0	TRUE	3	2	8	1.64
DP-Noise	6	0	TRUE	2	1	11	1.52
DP-Mask	14	4	TRUE	2	1	11	1.58

- **GIN.** For the TUDatasets benchmark, the backbone model comprises four GIN layers, each with a two-layer MLP. It utilizes a global mean pooling readout function, has a hidden size of 32, and adopts the ReLU activation function. Conversely, for the ogbg-molhiv dataset, the model consists of five GIN layers, a hidden size of 300, the ReLU activation function, and employs a global mean pooling for the readout function.

For all other hyper-parameter search space and training configurations of the experiments on the IMDB-B, IMDB-M, REDD-B, REDD-M5, and REDD-M12, we keep consistent with Han et al. (2022). For all other hyper-parameter search space and training configurations of the experiments on the PROTEIN, NCI1, and ogbg-hiv, we keep consistent with Ling et al. (2023). Note that instead of adopting the results of baseline methods on the ogbg-hiv dataset from the reference directly, we reported the results of rerunning the baseline experiments on the ogbg-hiv dataset, which is higher than the results in the reference.

Semi-supervised learning. We maintain consistency with You et al. (2020) for all hyper-parameter search spaces and training configurations. For all datasets, we conduct experiments at label rates of 1% (provided there are more than 10 samples for each class) and 10%. These experiments are performed five times, with each instance corresponding to a 10-fold evaluation. We report both the mean and standard deviation of the accuracies in percentages. During pre-training, we perform a grid search, tuning the learning rate among 0.01, 0.001, 0.0001 and the epoch number within 20, 40, 60, 80, 100.

Unsupervised representation learning. Following You et al. (2020); Lin et al. (2022), we use a 5-layer GIN as encoders. For all hyper-parameter search spaces and training settings in unsupervised learning, we also align with the configurations presented in You et al. (2020). We conduct experiments five times, with each iteration corresponding to a 10-fold evaluation. The reported results in Table 3 include both the mean and standard deviation of the accuracy percentages.

Transfer learning. Following the transfer learning setting in Hu et al. (2020b); You et al. (2020); Lin et al. (2022), we conduct graph classification experiments on a set of biological and chemical datasets via GIN models. Specifically, an encoder was first pre-trained on the large ZINC-2M chemical molecule dataset (Sterling & Irwin, 2015; Gómez-Bombarelli et al., 2018) and then was evaluated on small datasets from the same domains (i.e., BBBP, Tox21, ToxCast, SIDER, ClinTox, MUV, HIV, and BACE).

D MORE EXPERIMENTS RESULTS

Empirical studies. In Section 3, we investigate spectral alterations due to adding an edge in a toy graph (depicted in Figure 2a). The spectral changes are presented in Figure 2b. Further, in Figure 7, we analyze the consequences on the spectrum when edges from the same toy graph are removed. Notably, the removal of edges 0-4 and 3-4 results in minimal spectral variations. However, the removal of edges 5-6 and 6-7 leads to pronounced changes in both high and low frequencies, evident from the pronounced shifts in values λ_2 and λ_5 .

Hyperparameter sensitivities. We conducted experiments on the IMDB-BINARY dataset, leveraging various combinations of standard deviation σ and frequency ratio r_f for both low and high-frequency components to assess the impacts of DP-Noise parameters. For these experiments, we employed the GIN as our backbone model let σ and r_f from two search spaces, where $\sigma \in [0.1, 0.5, 1.0, 2.0]$ and $r_f \in [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8]$. We run 5 experiments and report the average values in Figure 8. Our observations indicate that introducing noise in the high-frequency components tends to enhance the test accuracy more markedly than when infused in the

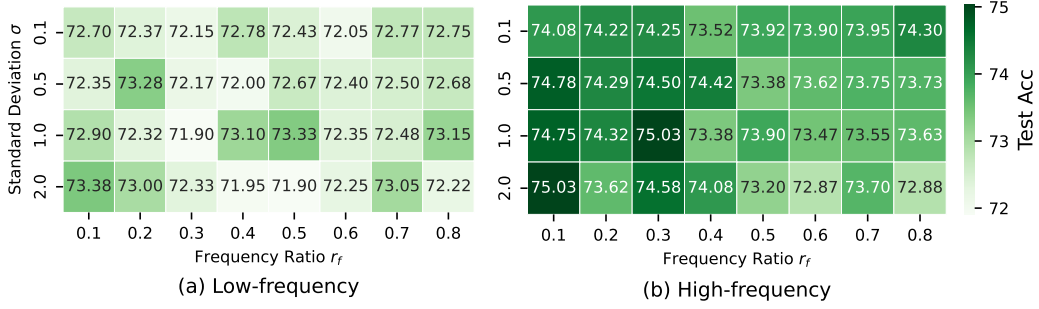


Figure 8: Effects of different hyperparameter combinations on the IMDB-BINARY dataset for graph classification via adding noise to (a) low-frequency and (b) high-frequency eigenvalues, respectively. The evaluation metric is accuracy.

low-frequency regions, as illustrated by the prevailing lighter color in Figure 8a. This observation resonates with the insights gleaned from Section 3. Building upon these general observations, we further elucidate the effects of each specific parameter below.

- **Effects of Standard Deviation σ .** For low-frequency components, the introduction of noise seems to not exhibit a consistent influence on accuracy. In contrast, when noise is applied to high-frequency components, we observe a discernible trend: accuracy tends to increase with increasing standard deviations. This suggests that the diversity introduced by elevating the standard deviation of noise can potentially bolster the classification performance of generated graphs.
- **Effects of Frequency Ratio r_f .** Similar to σ , for low-frequency components, increasing r_f does not consistently enhance or degrade accuracy across different standard deviations. On the other hand, in the high-frequency regime, a subtle trend emerges. As r_f increases, there is a nuanced shift in accuracy, suggesting that the spectrum of frequencies impacted by the noise has a nuanced interplay with the graph’s inherent structures and the subsequent classification performance.

E MORE DISCUSSIONS

Intuitions & Advantages of proposed methods. (1) *Properties Preservation.* Data augmentation should not only increase the quantity of training data but also enrich the quality of the learning experience for the model. Here, quality refers to the diversity, relevance, and realism of the augmented data. Therefore, property-retentive augmentations provide a more genuine learning context for the model, thus directly improving performance. (2) *Global Perspective.* Looking at the graph globally allows us to understand the larger structures and patterns within the graph. By making broader changes to the graph’s structure, global augmentations can create more diverse training instances, compared to local augmentations which might only create minor variations of the existing instances, therefore enhancing understanding of complex graph relationships and contributing to improved performance.

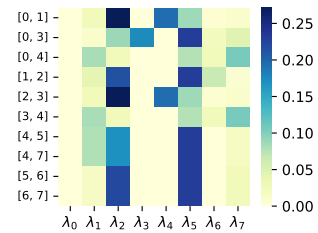


Figure 7: Absolute variation of eigenvalues when dropping different edges of the toy graph.

Broader Impact. Through a spectral lens, our Dual-Prism (DP) augmentation method presents both significant advancements and implications in the realm of graph-based learning. This can lead to improved performance, robustness, and generalizability of graph neural networks (GNNs) across a myriad of applications, from social network analysis to molecular biology. In addition, by utilizing spectral properties, our method provides a more transparent approach to augmentation. This can help researchers and practitioners better understand how alterations to graph structures impact learning outcomes, thereby aiding in the interpretability of graph data augmentation.

Limitations & Future Directions. A potential limitation of this study is its primary emphasis on homophily graphs. In contrast, heterophily graphs, where high-frequency information plays a more crucial role, are not extensively addressed (Bo et al., 2021). Looking ahead, it would be worth investigating learning strategies tailored to selectively alter eigenvalues, ensuring adaptability across diverse datasets.

Comparison with Existing Works. There are two related works about the spectral view on graph data augmentation, i.e., (Liu et al., 2022; Lin et al., 2022), while both are grounded in the GCL framework. Specifically, Liu et al. (2022) proposes a rule to find the optimal contrastive pair under the GCL framework instead of a novel augmentation method. GCL-SPAN Lin et al. (2022) centers on maximizing variance in the spectral domain, our observations indicate that overall spectral changes don’t always align with graph properties, as detailed in Section 3. Thus, constraining specific eigenvalues to remain invariant might be a more effective strategy for generating valid augmented graphs. Despite GCL-SPAN (Lin et al., 2022) also utilizing a spectral perspective, it in fact still modifies the spatial domain while optimizing in the spectral realm. In contrast, our techniques directly make alterations in the spectral domain, leading to more meaningful and effective alterations. This is evident by our methods’ superior performance in graph classification (see in Section 5.3) and underscores the efficiency of straightforward spectral modifications in creating more effective and discerning augmented graphs for graph classification.