Appendix of Submission #6158

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 1.

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

Table 2: 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

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, Shim, and Kang 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 and 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:

- 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 and 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, Chen, and Wang 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, Ying, and Leskovec 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.

Table 1: 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	✓		

- 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).

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 3.
- **Experiment of Figure 4.** In Figures 4a and 4b, labels in REDDIT-MULTI-12K for Class A and Class B are 1 and 10, respectively. The added edges in 4c are 3↔5, 3↔7, 1↔6, 2↔6, 0↔2, 1↔3, 1↔4, 2↔4, 4↔6, and 5↔7, respectively. The dropped edges in 4d are 0↔4, 3↔4, 4↔5, 4↔7, 0↔1, 2↔3, 0↔3, 5↔6, 6↔7, and 1↔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 3.** For both Figure 3a and 3b, we employ GIN as the backbone model and conduct experiments over five runs. In Figure 3b, 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 and 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

Table 3: 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

pooling readout function.

- **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, Chen, and Wang 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 4 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, Chen, and Wang 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 and 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).

More Experiments Results

Transfer learning. We draw comparisons between our methods and six baselines, including a reference model without pre-training (referred to *No-Pre-Train*), Informax (Veličković et al. 2018), EdgePred (Hamilton, Ying, and Leskovec 2017), AttrMasking (Hu et al. 2020b), ContextPred (Hu et al. 2020b) and GraphCL (You et al. 2020). The comparative results are shown in Table 4. Our methods demonstrated SOTA performance, outperforming competitors on half of the datasets. Especially, our methods consistently outperform the conventional GraphCL method, which indicates our data augmentation methods as better choices for graph contrastive learning. Notably, DP-Mask achieves an 83.52% ROC-AUC score on ClinTox, exceeding the performance of GraphCL by a substantial margin (nearly 10%). These findings demonstrate the enhanced efficacy of our techniques in the transfer learning setting for graph classification tasks.

Empirical studies. In Section , 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 1, 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. To further illustrate our observation, we present an additional example featuring a nine-node toy graph that also exhibits similar results, as shown in Figure 2.

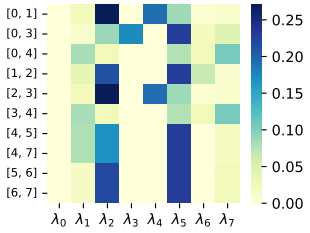
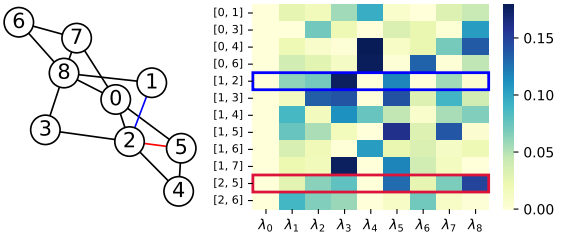


Figure 1: Absolute variation of eigenvalues when dropping different edges of the toy graph.

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

Table 4: Performance comparisons in the *transfer* learning setting. The best and second best results are highlighted with **bold** and underline, respectively. The metric is ROC-AUC scores (%). * and ** denote the improvement over the second-best baseline is statistically significant at levels 0.1 and 0.05, respectively. Baseline results are taken from (Hu et al. 2020b; You et al. 2020).

Dataset	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE
No-Pre-Train	65.80± 4.50	74.00± 0.80	63.40± 0.60	57.30± 1.60	58.00± 4.40	71.80± 2.50	75.30± 1.90	70.10± 5.40
Infomax	68.80± 0.80	75.30± 0.50	62.70± 0.40	58.40± 0.80	69.90± 3.00	75.30± 2.50	76.00± 0.70	75.90± 1.60
EdgePred	67.30± 2.40	76.00± 0.60	64.10± 0.60	60.40± 0.70	64.10± 3.70	74.10± 2.10	76.30± 1.00	**79.90± 0.90**
AttrMasking	64.30± 2.80	**76.70± 0.40**	**64.20± 0.50**	61.00± 0.70	71.80± 4.10	74.70± 1.40	77.20± 1.10	79.30± 1.60
ContextPred	68.00± 2.00	75.70± 0.70	63.90± 0.60	60.90± 0.60	65.90± 3.80	**75.80± 1.70**	77.30± 1.00	79.60± 1.20
GraphCL	69.68± 0.67	73.87± 0.66	62.40± 0.57	60.53± 0.88	75.99± 2.65	69.80± 2.66	78.47± 1.22	75.38± 1.44
DP-Noise	70.38± 0.91*	74.33± 0.42	64.08± 0.25	**61.52± 0.79**	76.26± 1.68	74.09± 2.30	78.63± 0.37	77.37± 1.76
DP-Mask	**71.63± 1.86****	75.51± 0.26	63.77± 0.22	61.33± 0.17	**83.52± 1.07****	75.02± 0.36	**78.73**± 0.77	79.45± 0.69



(a) Toy Graph (b) $\Delta\lambda$ of Toy Graph

Figure 2: (a) Another toy graph $\mathcal{G}'$ consisting of nine nodes. (b) Absolute variation in eigenvalues of $\mathcal{G}'$ when adding an edge at diverse positions. The red and blue rectangles represent when adding the corresponding edges in $\mathcal{G}'$ and the change of the eigenvalues.

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 3. Our observations indicate that introducing noise in the high-frequency components tends to enhance the test accuracy more markedly than when infused in the low-frequency regions, as illustrated by the prevailing lighter color in Figure 3a. This observation resonates with the insights gleaned from Section . 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.

We also conduct hyperparameter analysis in different learning settings. Figure 4 shows performances across multiple datasets, which reveals distinct trends in performance related to augmentation probability (*aug_prob*) and frequency ratio (*aug_freq_ratio*). Specifically, for the DD dataset, performance peaks with a high *aug_freq_ratio* and *aug_prob*, suggesting a preference for more frequent augmentations. In contrast, the MUTAG dataset shows optimal results at a lower frequency but higher probability, indicating a different augmentation response. The NCI1 dataset's best performance occurs at higher values of both parameters, while REDDIT-BINARY favors moderate to high frequency combined with a high probability, achieving its peak performance under these conditions. These patterns highlight the necessity of customizing hyperparameters to each dataset for optimal augmentation effectiveness.

Complexity and Time Analysis. Theoretically, the computational bottleneck of our data augmentation method primarily stems from the eigen-decomposition and reconstruction of the Laplacian matrix. For a graph with n nodes, the computational complexity of both operations is $O(n^3)$. In terms of implementation, we have measured the time cost required by our method. For each n, we randomly generated 100 graphs and recorded the average time and standard deviation required for our data augmentation method. The results, presented in Table 5, are measured in milliseconds. The average number of nodes in commonly used graph classification datasets is approximately between 10 and 500. Therefore, in the majority of practical training scenarios, the average time consumption of our algorithm for augmenting a single graph is roughly between 1 millisecond and 40 milliseconds. The experiments conducted here did not employ any parallel computing or acceleration methods. However, in actual training processes, it is common to parallelize data preprocessing using multiple workers or to precompute and store the eigen-decomposition results of training data. Therefore, the actual time consumption required for our method in implementations will be even lower. Therefore, **although the complexity is theoretically high, in fact, the time required is practically low**.

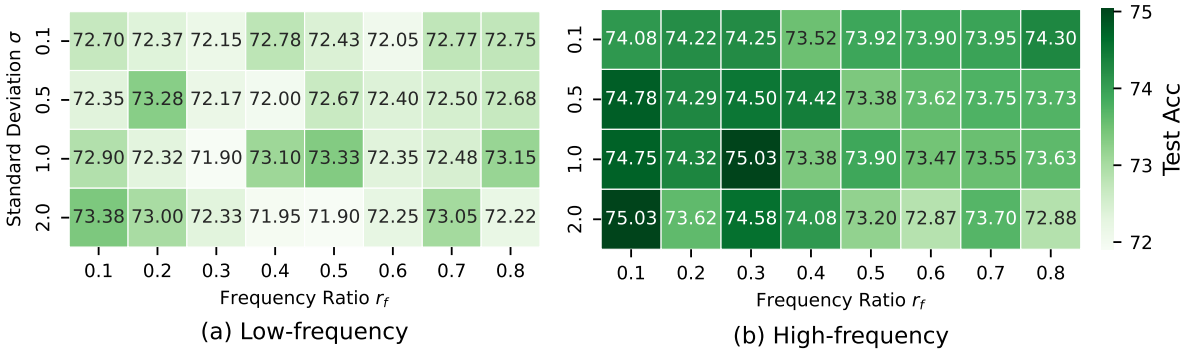


Figure 3: Effects of different hyperparameter combinations on the IMDB-BINARY dataset in the supervised learning setting for graph classification via adding noise to (a) low-frequency and (b) high-frequency eigenvalues, respectively. The evaluation metric is accuracy.

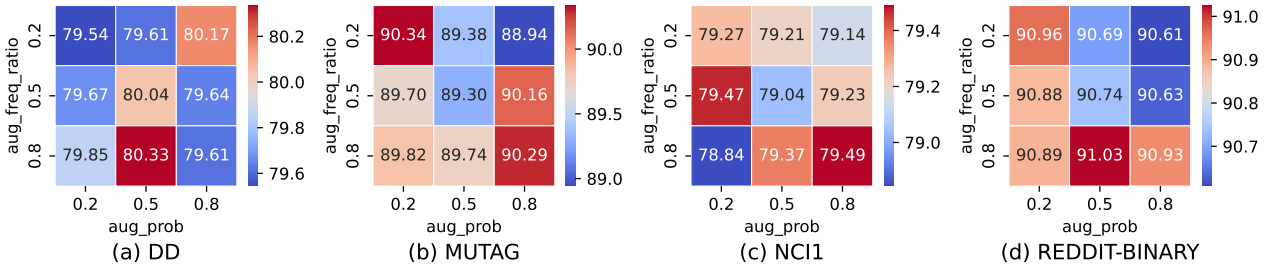


Figure 4: Effects of different hyperparameter combinations on different datasets in the unsupervised learning setting for graph classification via masking. The evaluation metric is accuracy.

More Discussions

Broader Impact. Through a spectral lens, our proposed 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. While there is a superficial overlap in terms of using "spectrum" for "graph data augmentation" with some related works, e.g., (Liu et al. 2022; Lin, Chen, and Wang 2022), the core methodologies and outcomes are distinctly different. *We believe it should be a new direction and open up new avenues for spectral-based augmentation techniques, allowing more diverse research on doing the augmentation on the spectrum.*

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). In addition, as we have discussed on the characteristics of molecular datasets above, in some situations, the high-frequency component may also include important information for graph classification tasks. Looking ahead, it would be worth investigating learning strategies tailored to selectively alter eigenvalues, ensuring adaptability across diverse datasets. This includes developing methods to safely create realistic augmented graphs and experimenting with mix-up techniques involving eigenvalues from different graphs.

High-Frequency Significance in Molecular Data. Incorporating insights from Table 4, our method shows varied performance across datasets. Specifically, within the Tox21 and ToxCast toxicology datasets, our approach does not achieve top-ranking results. This could be due to the potential significance of high-frequency components in capturing essential structural details within these datasets, which aligns with the intricate biological nature of the targets and stress response pathways represented in these datasets. The Tox21 dataset contains qualitative toxicity measurements for various biological targets, such as nuclear receptors and stress response pathways(Ma et al. 2022), which could be notably influenced by subtle molecular changes detected in the high-frequency domain. Consequently, it is plausible to suggest that high-frequency components play a vital role in distinguishing between toxic and non-toxic molecules by capturing subtle molecular differences crucial for predicting bioactivity and toxicity. In contrast to datasets focusing on

Table 5: Time cost required by our method. n: Number of nodes.

	$n = 10$	$n = 20$	$n = 100$	$n = 200$	$n = 500$	$n = 1000$
Time(ms)	0.76 ± 0.55	0.89 ± 0.43	2.50 ± 0.50	6.45 ± 0.58	41.35 ± 2.39	230.75 ± 8.76

broader properties like BBBP (Blood-Brain Barrier permeability) or BACE (inhibitor activity against beta-secretase), Tox21 and ToxCast encompass a wide range of bioassays that may be particularly sensitive to the high-frequency intricacies of molecular structures. Thus, the complexity of molecular compositions and the diversity of bioactivity assays in Tox21 and ToxCast imply that high-frequency components may contain essential information for toxicity prediction that may not be as pronounced in datasets such as BBBP.

Rationale for Choosing Graph Laplacian Decomposition. In our methodology, we chose to decompose the graph Laplacian L (where $L = D - A$) to do the perturbation and then reconstruct the graph. In terms of implementation, an alternative method can be directly decomposing A and perturbing its smaller eigenvalues. However, our motivation for this work is more on the inherent properties of graphs, and L offers a more nuanced reflection of these properties compared to A (Lutzeyer and Walden 2017). For future scenarios involving more complex disturbances to eigenvalues, leveraging the eigenvalues of L would be a more appropriate approach. In addition, our decision to decompose L also follows general spectral graph convolution methodologies (Kipf and Welling 2016a).

Loss-pass Filtering of GNNs vs. DP Augmentation. We then discuss the relationships and differences between the loss-pass filtering of GNNs, caused by the model's neighborhood aggregation processes (Nt and Maehara 2019; Kipf and Welling 2017), and our proposed methods.

- *Relationship*. The low-pass effect of GNNs exactly aligns with our motivation - increasing the diversity of data (i.e. altering the high-frequency part) without misguiding the classification (i.e. preserving the low-frequency part). In other words, because DP augmentation enhances the training-data diversity by introducing randomness to the high-frequency spectrum, the low-pass effect of GNNs preserves the core attributes necessary for correct classification, ensuring label consistency across original and augmented data. By using this natural smoothing capability, our method introduces label-consistent variations to the data. This strategy boosts model robustness and generalization without compromising classification accuracy, affirming the efficacy and theoretical integrity of our method.
- *Difference*. From the signal processing perspective, our method is more like introducing random perturbations to high-frequency components rather than acting as another low-pass filter. For better intuitive understanding, take the image processing as an example, a low-pass filter achieves image denoising, while random perturbation of high-frequency noise sharpens images. These two effects are different and even have a certain degree of opposition. Due to such difference, these two operations can be used jointly during the image processing - in our work, we first use augmentation to randomly perturb the high-frequency part, and then pass it through the low pass filter of GNNs.

Possible reasons of why DP-Noise outperforms DP-Mask. DP-Noise applies subtle, controlled noise to high-frequency eigenvalues, preserving essential low-frequency components while introducing diversity through minor spectral changes. This balance allows DP-Noise to produce augmented graphs that are both diverse and stable, enhancing model generalization and robustness. Conversely, DP-Mask removes selected high-frequency eigenvalues outright, creating more pronounced structural shifts. While effective in diversifying, this approach can introduce instability by over-modifying spectral components, impacting performance consistency.

References

Adhikari, B.; Zhang, Y.; Ramakrishnan, N.; and Prakash, B. A. 2018. Sub2vec: Feature learning for subgraphs. In *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part II 22*, 170–182. Springer.

Bo, D.; Wang, X.; Shi, C.; and Shen, H. 2021. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 3950–3957.

Feng, W.; Zhang, J.; Dong, Y.; Han, Y.; Luan, H.; Xu, Q.; Yang, Q.; Kharlamov, E.; and Tang, J. 2020. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33: 22092–22103.

Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; and Aspuru-Guzik, A. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2): 268–276.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Han, X.; Jiang, Z.; Liu, N.; and Hu, X. 2022. G-mixup: Graph data augmentation for graph classification. In *International Conference on Machine Learning*, 8230–8248. PMLR.

Hassani, K.; and Khasahmadi, A. H. 2020. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, 4116–4126. PMLR.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020a. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33: 22118–22133.

Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; and Leskovec, J. 2020b. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*.

Kipf, T. N.; and Welling, M. 2016a. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kipf, T. N.; and Welling, M. 2016b. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *In International Conference on Learning Representations*.

Lin, L.; Chen, J.; and Wang, H. 2022. Spectral Augmentation for Self-Supervised Learning on Graphs. In *The Eleventh International Conference on Learning Representations*.

Ling, H.; Jiang, Z.; Liu, M.; Ji, S.; and Zou, N. 2023. Graph Mixup with Soft Alignments. *arXiv preprint arXiv:2306.06788*.

Liu, N.; Wang, X.; Bo, D.; Shi, C.; and Pei, J. 2022. Revisiting graph contrastive learning from the perspective of graph spectrum. *Advances in Neural Information Processing Systems*, 35: 2972–2983.

Lutzeyer, J.; and Walden, A. 2017. Comparing graph spectra of adjacency and laplacian matrices. *arXiv preprint arXiv:1712.03769*.

Ma, H.; Bian, Y.; Rong, Y.; Huang, W.; Xu, T.; Xie, W.; Ye, G.; and Huang, J. 2022. Cross-dependent graph neural networks for molecular property prediction. *Bioinformatics*, 38(7): 2003–2009.

Morris, C.; Kriege, N. M.; Bause, F.; Kersting, K.; Mutzel, P.; and Neumann, M. 2020. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*.

Narayanan, A.; Chandramohan, M.; Venkatesan, R.; Chen, L.; Liu, Y.; and Jaiswal, S. 2017. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*.

Nt, H.; and Maehara, T. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2019. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*.

Sterling, T.; and Irwin, J. J. 2015. ZINC 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11): 2324–2337.

Sun, F.-Y.; Hoffman, J.; Verma, V.; and Tang, J. 2019. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *International Conference on Learning Representations*.

Suresh, S.; Li, P.; Hao, C.; and Neville, J. 2021. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems*, 34: 15920–15933.

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2018. Deep Graph Infomax. In *International Conference on Learning Representations*.

Verma, V.; Lamb, A.; Beckham, C.; Najafi, A.; Mitliagkas, I.; Lopez-Paz, D.; and Bengio, Y. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, 6438–6447. PMLR.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*.

Yoo, J.; Shim, S.; and Kang, U. 2022. Model-agnostic augmentation for accurate graph classification. In *Proceedings of the ACM Web Conference 2022*, 1281–1291.

You, Y.; Chen, T.; Shen, Y.; and Wang, Z. 2021. Graph contrastive learning automated. In *International Conference on Machine Learning*, 12121–12132. PMLR.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33: 5812–5823.