Appendix for Attribute-guided Dynamic Prompt Learning for Graph Neural Networks

Anonymous submission

A Dataset Statistics

We conduct experiments on real-world benchmark datasets, including Cora, Citeseer, Amazon-Computer (Amz-Comp.), Amazon-Photo (Amz-Photo). Cora and Citeseer (Kipf and Welling 2017) are representative citation datasets, where nodes represent papers, and edges denote citation relationships between papers. Amazon-Computer (A-Comp.) and Amazon-Photo (A-Photo) (Shchur et al. 2018) are co-purchase network datasets, nodes signify products, and an edge between two nodes suggests that these products are often bought together. The statistics of these datasets are summarized in Table 1.

Dataset	# Nodes	# Edges	# Features	# Classes
Cora	2,485	5,069	1,433	7
Citeseer	2,110	3,668	3,703	6
A-Photo	7,487	119,043	745	8
A-Comp.	13,381	245,778	767	10

Table 1: Dataset Statistics.

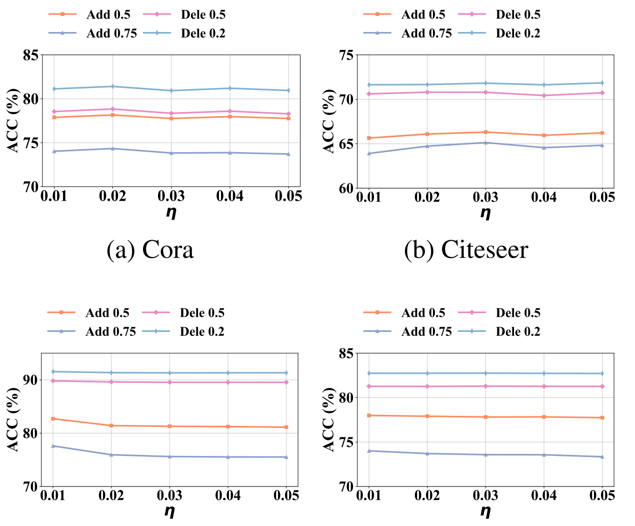
B Sensitivity Analysis

B.1 Hyper-parameters for Adversarial Graph Augmentations

We investigate the impact of the hyper-parameter η, which controls the step size for adversarial graph augmentations. η is selected from 0.01 to 0.05 with step 0.01. The results are shown in Figure 1. As observed, the accuracy (ACC) under different perturbation settings remains relatively stable. When η is larger than 0.02, the changes in ACC become negligible. In the experiments presented in the previous subsections, we choose the optimal values of η that yield the highest ACC for each dataset. Specifically, we set η to 0.02 for Cora and Citeseer, A-Computers, and 0.01 for A-Photo and A-Comp.

B.2 Hyper-parameters for KNN graph construction

Figure 2 shows the impact of the hyper-parameter k used for KNN graph construction, which is selected from $\{10, 20, 30, 40, 50, 60\}$. In Figure 2, the performance of our



(a) Cora (b) Citeseer

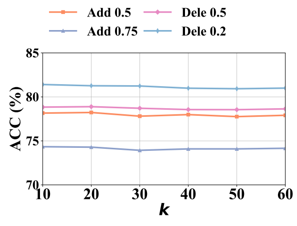
(c) A-Photo (d) A-Comp.

Figure 1: The sensitivity analysis of hyper-parameters used in the adversarial graph perturbation module.

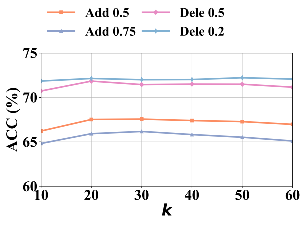
model remains relatively stable across a wide range of k values, demonstrating the robustness of our model to this hyper-parameter. However, a slight decline in accuracy can be observed when k exceeds 30. This suggests that a larger k may introduce noisy or less relevant neighbors, potentially weakening the quality of the constructed attribute graph. Based on empirical observations, we set $k = 10$ for Cora, A-Photo, and A-Computers, and $k = 20$ for Citeseer.

C Performance Comparison with Different Graph Perturbation

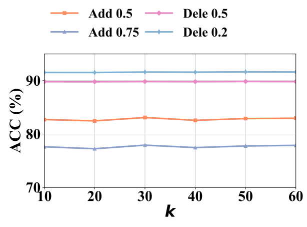
In this experiment, we compare the performance of the baseline GCN and our method across all datasets. We corrupt the original graph by adding or deleting edges, which the perturbation ratio is selected from 0.75 to 0 with a step size of 0.25. As illustrated in Figure 3, our method consistently outperforms the vanilla GCN in all perturbation ratios. On Cora and Citeseer, our method achieves a performance im-



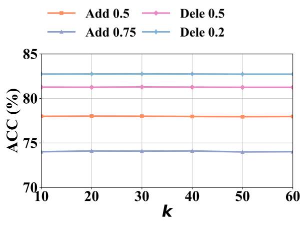
(a) Cora



(b) Citeseer

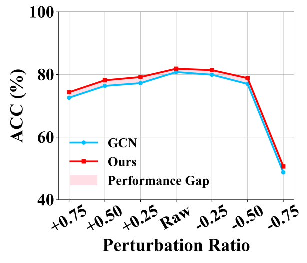


(c) A-Photo

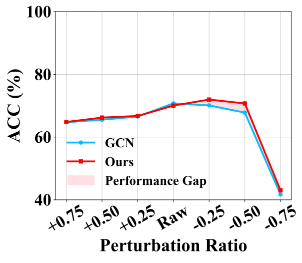


(d) A-Comp.

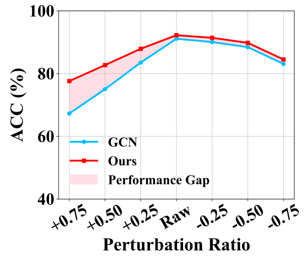
Figure 2: The sensitivity analysis of hyper-parameters k.



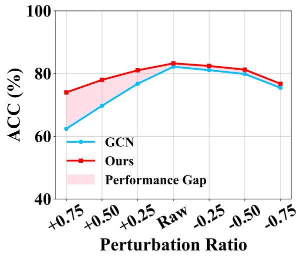
(a) Citeseer



(b) Citeseer



(c) A-Photo



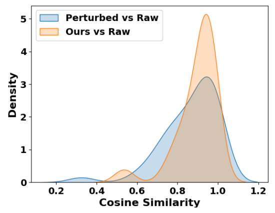
(d) A-Comp.

Figure 3: Performance comparison on graphs with different perturbation ratios.

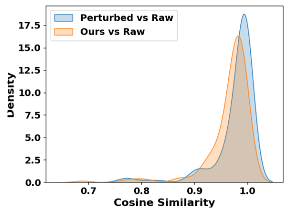
provement in both edge deletion and addition settings. On A-Photo and A-Comp, both GCN and our model perform similarly under clean or mildly perturbed graphs. However, as the perturbation ratio increases, our model shows clear advantages. Notably, our method maintains high accuracy even under severe perturbations (adding 0.75 edges), where GCN experiences a drastic performance drop. These results highlight the effectiveness of our method in preserving performance across different datasets and perturbation levels.

D Representation Consistency under Perturbation

To provide further comparison, we provide cosine similarity distributions on the other datasets. The results in Figure 4 show the same conclusions as before.



(a) Citeseer



(b) A-Comp.

Figure 4: Cosine similarity between original representations and perturbed/prompt-based representations.

E Analysis of Training and Testing Time

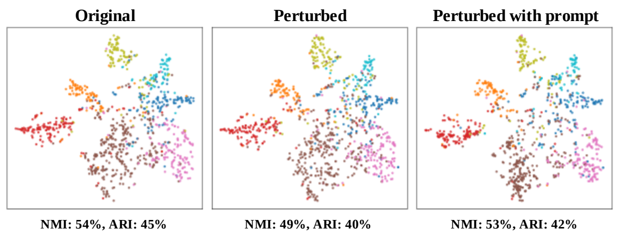
Table 2 reports the training and testing time per epoch for various models. Our model achieves a good balance between efficiency and performance. Although it incurs moderately higher cost than GCN due to dynamic prompt generation, our model remains significantly more efficient than EERM and CITGCN, which incur high computational costs from complex components such as multiple environments and clustering strategies. In contrast, StruRW and CaNet run faster than GCN due to their lightweight architectures. GOOD-AT exhibits significantly slower training due to generating perturbed graphs on the CPU, which substantially slows down the overall process. Overall, the results demonstrate that our model maintains competitive accuracy while ensuring scalability under structural perturbations.

	GCN	EERM	CITGCN	StruRW	CaNet	GOOD-AT	Ours
Train	15.86	258.38	35.13	9.69	13.91	61628.79	21.88
Test	1.87	2.46	2.21	2.59	1.18	775.37	2.32

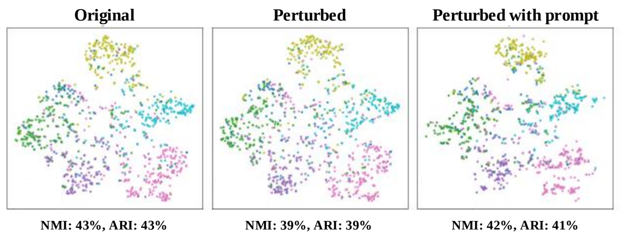
Table 2: Training and inference time comparison (ms).

F Visualization

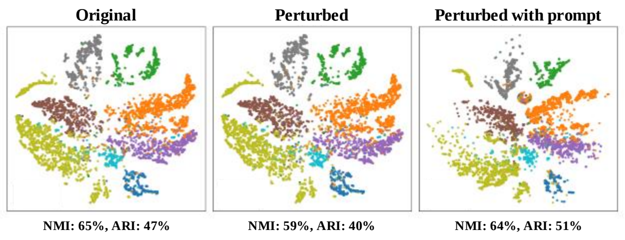
We visualize the learned representations on the test set using t-SNE (Van der Maaten and Hinton 2008). During visualization, nodes with the same ground-truth labels are shown in the same color. This allows us to assess the discrimination of the learned representations through the compactness of same-class nodes. To further quantify this, we perform clustering on the representations and report the Adjusted Rand Index (ARI) (Hubert and Arabie 1985) and Normalized Mutual Information (NMI) (Strehl and Ghosh 2002). Figure 5 illustrates the representations learned from the graph after removing 50% of the existing edges The results indicate that perturbations lead to less compact clusters, resulting in lower ARI and NMI scores compared to the original graph. These results confirm the presence of representation deviation caused by structural shifts. By introducing prompt vectors, our method alleviates this deviation and produces more discriminative representations, demonstrating improved robustness under structural perturbations. Due to the large number of nodes and edges in the A-Computers dataset, the t-SNE computation becomes prohibitively expensive. Therefore, we omit the visualization for this dataset.



(a) Cora



(b) Citeseer



(c) A-Photo

Figure 5: T-SNE visualization of node representations from the original graph, the perturbed graph, and the perturbed graph with our prompt

G Experimental Environment and Equipment

All experiments are implemented using PyTorch and conducted on an NVIDIA RTX 3090 GPU with 24 GB of memory and an 11th Gen Intel(R) Core(TM) i9-11900K CPU running at 3.50GHz.

References

Hubert, L.; and Arabie, P. 1985. Comparing partitions. *Journal of classification*, 2: 193–218.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of Graph Neural Network Evaluation. *ArXiv*.

Strehl, A.; and Ghosh, J. 2002. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec): 583–617.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).