Improving Graph Representation Learning with Augmentations, Uncertainty Quantification and Large Language Model Guidance

by

Puja Trivedi

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2024

Puja Trivedi

pujat@umich.edu

ORCID iD: 0000-0003-1874-8992

DEDICATION

To my family.

ACKNOWLEDGEMENTS

The PhD journey is often long and challenging, and I feel incredibly lucky to have been supported by many people who have made this path brighter. While I may not name everyone here, please know that I am forever grateful!

First, I want to extend my deepest gratitude to my wonderful advisor, Prof. Danai Koutra, who taught me how to conduct research and become an independent scholar. Her patience and kindness were invaluable as she guided me through research setbacks and helped me explore new areas. I am further indebted for her technical guidance as we conducted projects, and professional advice as I navigated the job market; both were invaluable. I feel extremely lucky to have worked with such a dedicated and caring advisor, and hope that I can follow in her example. I am also grateful to my "co-advisor," Dr. Jay Thiagarajan. Prior to working with Jay in 2021, I had been facing some project setbacks, and found myself losing motivation. Jay's sheer enthusiasm, optimism and kindness helped me rediscover the fun of doing research again, for which I am incredibly thankful. Under his guidance, I learned how to pursue questions that matter, believe in my ideas and challenge myself with new areas. I am also grateful for his advice throughout projects, as well as his support during last-minute paper rushes. I would also like to thank my committee members, Prof. Maggie Makar and Prof. Angela Violi, for their valuable feedback and insightful questions. Thank you for taking the time for the detailed comments, which have strengthened the quality of this work.

Throughout my studies, I was fortunate to gain further practical experience through internships at Lawrence Livermore National Laboratory, Adobe Research, and Amazon, where I was mentored by many outstanding professionals. In particular, I want to thank Mark Heimann, Rushil Anirudh, Ryan A. Rossi, Nurendra Choudhary, Vassilis Ioannidis, and Eddie Huang for their mentorship.

I'd also like to give a big thank you to my lab mates, Caleb Belth, Alican Buyukcakir, Marlena Duda, Mark Heimann, Di Jin, Donald Loveland, Fatemeh Vahedian, Yujun Yan, Jing Zhu, Tara Safavi, and Jiong Zhu, who helped me refine ideas, improve talks, and navigate the job market. I learned so much from each of you and appreciated the camaraderie we shared.

I am also extremely grateful to my talented collaborators, Nurendra Choudhary, David Arbour, Nesreen K. Ahmed, Rushil Anirudh, Jayaraman J. Thiagarajan, Alican Büyükçakır, Ekdeep Singh Lubana, Danai Koutra, Franck Dernoncourt, Robert P. Dick, Edward W. Huang, Mark Heimann, Vassilis Ioannidis, Di Jin, Sungchul Kim, Yin Lin, Ruiyu Li, Nedim Lipka, Vivek Narayanaswamy, Namyong Park, Yinlong Qian, Ryan A. Rossi, Karthik Subbian, Fatemeh Vahedian, Yu Wang, Yujun Yan, and Yaoqing Yang. None of this work would have been possible without your invaluable support. I hope that we may continue to work together in the future!

My interest in a PhD began as an undergraduate due to the support of the Meyerhoff Scholars Program. I am appreciative of the programs support and guidance. I would also like to acknowledge mentors from my undergraduate research experiences, including Talmo Pereira, John Winder, John Boaz Lee, Marie desJardins Park, Robert Goldman, and Jonathan Pillow, whose early mentorship helped encourage me to pursue this path. Mentoring undergraduates can be a thankless task, and I appreciate their generosity. Notably, this dissertation marks my 21st year as a professional student, and I am deeply appreciative of all my teachers, professors for the quality education I received through the public school system.

To my excellent friends, Jiong Zhu, Yin Lin, Megha Ghosh, Karan Desai, Dandan Shan, Sangeeta Rao, Preeti Ramaraj, Karthick Krishnan, Swati Adipudi and many more, thank you for your friendship and support along this journey. I appreciated hanging out, laughing and chilling with all of you.

Finally, I extend my deepest gratitude to my family. To my father, Bhagirath Trivedi, for teaching me the value of hard work and encouraging me to pursue my interests. To my mother, Tejal Trivedi, for taking care of us and for making sure I was taking care of myself, eating, and happy while pursuing my studies. To my brother, Arjun Trivedi, thank you for being my best friend, listening to my work-related frustrations, and for always laughing at my jokes. Lastly, a very special thank you to my partner, Ekdeep Singh Lubana. I cannot overstate how important his love, patience, and unwavering support were in helping me complete this PhD. Each time I felt discouraged, he was there, cheering me on and giving me confidence. For every success, he celebrated alongside me and made them more special. Ekdeep is truly the most kind and generous person I know, and I am grateful for all the happiness and fun we share.

Financial support for this work was provided by National Science Foundation under CAREER Grant No. IIS 1845491, Grant No. IIS 2212143, Army Research Awards, and LLNL Subcontract B651167.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

ABSTRACT

Expressive graph representation learning is important to many high-impact applications as structured data across many domains can be naturally represented using graphs. While the advent of graph neural networks has led to considerable success on a variety of graph-based tasks, there remains room for improvement when the quality of GNN representations is determined with respect to not only in-distribution task performance but also other desiderata, such as generalization under distribution shift, trust-worthiness, or robustness. To this end, this thesis is broadly interested in understanding and improving the quality of graph representations *beyond accuracy* and makes contributions to several of the aforementioned desiderata.

The first part of this thesis considers the generalization and expressiveness of graph representations learnt using contrastive learning, as expressivity is often a prerequisite for other model desiderata. In particular, we study the role of augmentations, identifying several weaknesses of popular, generic graph augmentations strategies and derive a corresponding generalization bound. In the second part, we focus on improving the reliability of uncertainty estimates when performing predictive tasks, as expressivity alone is not sufficient for safe model deployment. To this end, we propose a lightweight training protocol that improves estimate quality, even under challenging distribution shift settings. The final part of this thesis considers strategies for leveraging large language models to further enhance GNN performance and robustness, particularly by supporting capabilities that are not realizable from only the GNN or structured data alone. Specifically, we propose a framework for LLM guided text-attributed graph clustering that uses the LLM to disambiguate uncertain nodes, and improves zero-shot clustering capabilities for disconnected nodes. Finally, we study the robustness of joint LLM and GNN models to structural and text-based adversarial attacks.

CHAPTER 1

Introduction

Relational data, which model interactions or relations amongst entities, are prevalent across many domains, and can be naturally represented using networks or graphs. Indeed, high impact applications such as drug-discovery [20], algorithmic reasoning [21] and making recommendations [22, 23], have been successfully formulated as machine learning tasks on graph data. Success in such tasks has primarily been buoyed by the representation learning capabilities of graph neural networks (GNNs) [24, 25, 26, 27], a class of neural architectures expressively designed to capture graph-specific inductive biases on discrete, variable-sized inputs.

While GNNs have rapidly become the defacto standard on a variety of graph-based tasks, e.g., node classification [28], graph classification [29], link prediction [22], graph clustering [30] and graph alignment [31], there remains considerable room for improvement as we evaluate the quality of learnt representations beyond task performance ("accuracy"). Indeed, much like their vision counterparts [32, 33, 34], GNNs are weak generalizers under distribution shift [18, 35], susceptible to adversarial attacks [36, 37], biased during decision making [38, 39, 40] and often poorly calibrated [41, 42]. Notably, practical deployment in high-risk scenarios requires strong performance on these desiderata, in addition to merely achieving strong task performance. Thus, there has been growing interest in the graph machine learning community to pursue algorithmic [43, 44] and architecture based [45] strategies to pursue these aims, where challenges can arise from limitations in model expressivity [27, 46], training paradigms [47, 48], limited data [49], class imbalance [50] and amongst other sources.

In Parts 1 and 2 of this thesis, we focus on two of these desiderata, namely: understanding representation expressivity, which is arguably a prerequisite to others, and improving calibration. Specifically, in Part I, we rigorously study the limitations of graph contrastive learning (GCL) [1, 51, 52, 53], as a training protocol. We choose to focus on GCL as CL in vision and other modalities has not only lead to state-of-the-art downstream task performance [54, 55, 56], it has been shown to have improved the robustness [57, 58], semantic consistency [59] and transferability [60] relative to supervised counterparts. By identifying and rectifying limitations in GCL, we can better support these properties in the finetuned or downstream GNN-based tasks. In Part II, we focus on the complementary objective of obtaining reliable uncertainty estimates on three foundational GNN-based tasks. Reliable estimates can aid model trust-worthiness by improving calibration [34], generalization gap prediction [61], out-of-distribution detection [62] and other tasks dependent [63] on accurate uncertainty quantification, improving overall model quality.

While Parts Iand IIfocus on algorithmic, GNN-based strategies for holistically improving GNN performance, the recent, unprecedented success of large language models (LLMs) [64, 65] has enabled an alternative paradigm for GNN-based tasks where LLM world knowledge and reasoning capabilities complement GNN capabilities. Indeed, recent work demonstrates that various strategies e.g., co-training, pretraining, finetuning, prompting [66, 67, 68, 69, 70, 71, 72, 73, 74], for utilizing LLMs in conjunction with GNNs can improve supervised task performance on text-attributed graphs [75, 76] i.e., graphs with natural language text as node information. However, it remains, to the best of our knowledge, under-explored how these joint LLM+GNN methods perform on other important factors of task performance (calibration, robustness, fairness, etc).

To this end, Part IIIof this thesis focuses on introducing new capabilities that were previously inaccessible when relying upon only structural data and GNNs by using both LLMs and GNNs in the prediction pipeline. In particular, we first consider how LLM world knowledge can be leveraged to reduce uncertainty, improve performance, and support better zero-shot performance when performing graph clustering [77, 78, 30].

1.1 Contributions

Here, we discuss our contributions in more detail and summarize them in Table 1.1. While this thesis is broadly focused on the improving the performance on graph based tasks beyond only accuracy or naive task performance, we focus on three particular sub-areas in the following parts. In Part I, we focus on better understanding representation expressivity by studying the role of augmentations in graph contrastive learning. Part IIfocuses on obtaining reliable uncertainty estimates on various supervised graph-based tasks, leading to better model trust-worthiness. Lastly, in Part III, we combine our insights from the preceding sections, to propose a novel framework for LLM guided graph clustering. We further study how to jointly combine LLMs and GNNs for graph based tasks in order to improve performance on user-specified objectives, such as reducing reliance upon sensitive features when making predictions.

Table 1.1: **Overview of contributions.** This thesis consists of three parts, broadly interested in improving the performance of GNNs beyond accuracy.

Aim	**Task(s)**	**tl;dr**	**Venue**	**Ch.**
Augmentations in Graph CL	Graph Classification	Better Practices for Graph CL	WWW21 [79]	3
	Graph Classification	Generalization Analysis of Graph CL	NeurIPS22 [80]	4
Improved Uncertainty Estimations with GNNs	Graph/Node Classification	Flexible Uncertainty Estimation for GNN Classifiers	ICLR 24 [81]	5
	Link Prediction	Better Calibration for Link Predictors	ICASSP24[82]	6
Combining World and Structural Knowledge	Graph Clustering	LLM Guided Graph Clustering	In Submission	7
	Node Classification	Exploring Robustness of LLM + GNN Models	In Preparation	8

Qualities of Representations from Graph Self Supervised Learning. In the first part of this thesis, we rigorously study the role of augmentations when seeking to obtain expressive representations using graph contrastive learning. We contribute the following:

- **Better Practices for Graph CL Augmentations**: We probe the quality of representations learnt by popular graph CL frameworks using generic graph augmentations and find that such augmentations can destroy task-relevant information as well as harm the model's ability to learn discriminative representations. Based on our findings, we propose several sanity checks that enable practitioners to quickly assess the quality of their model's learned representations. This work was published in **WWW 2021**.

- **Analysis of Data-Centric Properties for Graph Contrastive Learning**: We perform a generalization analysis for CL when using generic graph augmentations, with a focus on data-centric properties, specifically *invariance* to task-irrelevant semantics, *separability* of classes in some latent space, and *recoverability* of labels from augmented samples. Our theory motivates a synthetic data generation process that enables control over task-relevant information with pre-defined optimal augmentations, enabling further insights into automated methods. This work was published in **NeurIPS 2022**.

Uncertainty in GNN Based Tasks In the second part of this thesis, we focus on improving the reliability of uncertainty estimates obtained from GNNs on supervised, predictive tasks. Our contributions include:

- **Accurate Estimation of Epistemic Uncertainty for Graph Classification and Node Classification**: We propose G-ΔUQ, a novel training framework designed to improve intrinsic GNN uncertainty estimates. Through extensive evaluation under covariate, concept and graph size shifts, we show that G-ΔUQ leads to better calibrated GNNs for node and graph classification. It also improves performance on the uncertainty-based tasks of out-of-distribution detection and generalization gap estimation. This work was published in **ICLR 2024**.

- **Improving Link Prediction Calibration**: We propose E-ΔUQ, an architecture-agnostic framework designed for improving link prediction calibration with minimal overhead. We further demonstrate the importance of considering node-level uncertainties when estimating link uncertainty, which despite its importance, had been overlooked. This work was published in **ICASSP 2024**.

Combining World and Structural Knowledge in Graph Representation Learning. In the final part of this thesis, we consider how LLMs can be used to augment the performance of GNNs. We contribute the following:

- **LLM Guided Graph Clustering**: We propose an active learning framework that performs **g**raph **c**lustering using **L**LM **r**efinment (GCLR) by selectively prompting an imperfect LLM oracle for feedback and, subsequently, finetuning the GNN-based clustering solution to incorporate the feedback. **GCLR** uses different prompting strategies to improve the LLM's reliability as an oracle and uses noise-controlling fine-tuning to handle this imperfect, but useful feedback. Extensive experiments demonstrate that GCLR can significantly improve clustering performance over state-of-the-art GNN methods.

- **Understanding Robustness of LLM+GNN Models on Text Attributed Graphs** In this chapter, we seek to understand the robustness of joint LLM+GNN models when performing node classification. Namely, we evaluate the sensitivity of two popular classes of joint models on structural, text-adversarial attacks, and semantic preserving natural text perturbations. Our analysis helps understand the potential for attack transferability across modalities and vulnerabilities present in this new class of models.

CHAPTER 2

Preliminaries

In this chapter, we quickly review some preliminaries on graph machine learning and some shared notations. We will introduce specific background and notations in the corresponding chapter as needed.

2.1 Graphs

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a mathematical structure used to capture relationships between entities. Here, $\mathcal{V}$ represents the set of nodes (or vertices), and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges connecting pairs of nodes. For example, in social networks, nodes may correspond to users and edges may correspond to interactions between users, or, in molecular graphs, nodes may correspond to different atoms, and edges to bonds. Graphs may be **directed**, where edges have orientations (i.e., $(u, v) \in \mathcal{E} \neq (v, u) \in \mathcal{E}$), or **undirected**, where edges are bidirectional (i.e., $(u, v) = (v, u)$). Graphs may optionally include nodes and/or edges attributes that can be represented as feature vectors and/or natural language text. Formally, let $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ denote the node feature matrix, where each row $\mathbf{x}_v \in \mathbb{R}^d$ represents the d-dimensional features of node v. Similarly, edge features can be represented by a matrix $\mathbf{E}$ where each entry $\mathbf{e}_{(u,v)}$ contains the features for edge (u, v). Modern graph learning techniques seek to effectively leverage both attributes and structure to succeed on various machine learning tasks, which we introduce below.

2.2 Graph Machine Learning Tasks

Graphs can be naturally used to represent structured information in a variety of domains, including social networks analysis [83, 84, 85], bioinformatics [21], computer vision [86], and recommendation systems [87]. Representing such information as a graph makes it amenable to performing a number of useful machine learning tasks. Here, we assume without loss of generality that we are working with an unweighted, undirected, attributed graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, and are provided optional labels $\mathcal{Y}$, corresponding to the underlying task. We further assume that the objective is to learn a function $f : \mathcal{G} \to \mathcal{Y}$ that maps from graphs (or nodes within graphs) to the label space $\mathcal{Y}$ appropriate for the given task.

2.2.1 Node Classification

In node classification, the goal is to predict a label $y_v \in \mathcal{Y}$ for each node $v \in \mathcal{V}$, given a partially labeled graph. For a labeled node subset $\mathcal{V}_L \subset \mathcal{V}$ with labels $\{y_v\}_{v \in \mathcal{V}_L}$, the task involves learning a function $f : \mathcal{G} \to \mathcal{Y}$ that assigns labels to the remaining nodes $\mathcal{V} \setminus \mathcal{V}_L$, where the model is trained to minimize the loss on the labeled subset with the expectation f will non-trivially generalize to the unlabeled portion. Here, f is designed to utilize both the node features $\mathbf{X}$ and the graph structure E for inference. Node classification has widespread applications, such as predicting user attributes in social networks or classifying academic papers in citation networks.

2.2.2 Graph Classification

In graph classification, the objective is learn a function, f from a training set of graphs, $\mathbf{G} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n\}$ and graph-level labels, $y_1, y_2 \ldots y_n$, such that f can be used to predict the label of unseen graphs at inference time. Formally, we train $f : (\mathcal{G}) \to y$ on the labeled training data to minimize a classification loss with the expectation f will generalize to other graphs in the distribution at inference time. Graph classification is commonly applied in areas like chemistry and biology, where each molecule (represented as a graph) is classified according to properties like toxicity or activity.

2.2.3 Graph Clustering

Graph clustering aims to identify communities or groups of similar nodes within a graph, generally without label supervision. Namely, given $\mathcal{G}$, the objective is to partition $\mathcal{V}$ into k clusters $\{C_1, C_2, \ldots, C_k\}$, such that nodes within each cluster are more similar or semantically related than nodes in different clusters. Formally, clustering can be considered as learning a mapping $f : \mathcal{G} \to \{1, \ldots, k\}$ where f is trained in an unsupervised fashion. Notably, graph clustering has applications in community detection in social networks, functional grouping in biological networks, and anomaly detection.

2.3 Graph Neural Networks (GNNs)

To perform the aforementioned tasks, graph neural networks (GNNs) have emerged as the state-of-the-art architectures as they are able to utilize both graph structure and attributes to learn expressive representations. Moreover, unlike traditional neural networks, which often cannot handle discrete, non-euclidean, variable sized data, GNNs utilize message-passing or neighborhood aggregation to capture the dependency structure of graphs and scale to large networks.

At a high level, GNNs learn node representations by iteratively updating aggregated information from its neighbors. This process can be described by a function $h : \mathcal{V} \rightarrow \mathbb{R}^d$ that computes node embeddings as follows:

$$\mathbf{h}_v^{(k)} = \text{AGGREGATE}^{(k)} \left(\left\{ \mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v) \right\} \right)$$

where $\mathcal{N}(v)$ represents the neighbors of v, and AGGREGATE$^{(k)}$ is a task-specific function that combines information from node v's neighbors at layer k. After stacking multiple message passing layers, an optional READOUT layer can be used to obtain a graph level representation, and an MLP layer or light-weight classifier can be added to perform the tasks mentioned above. Various GNN architectures have been proposed to refine the original graph convolutional network [24] that provide improved aggregation, sampling, positional and distance awareness, amongst other improvements [26, 28, 25, 88]. Indeed, developing better architectures remains an active and exciting research area, and we use several modern architectures througout this dissertation.

Part I: Improving Augmentations in Graph Contrastive Learning

CHAPTER 3

Better Practices for Graph CL Augmentations

3.1 Introduction

Graph neural networks (GNNs) have been successfully used to learn representations for various supervised or semi-supervised graph-based tasks, including graph-based similarity search for web documents [89], fake news detection through propagation pattern classification [90, 84, 83, 85], activity analysis in web and social networks (e.g., discussion threads on Reddit, code repository networks on Github) [91], and scientific graph classification [92, 93, 94]. However, in many practical scenarios, labels are scarce or difficult to obtain. For example, web pages are seldom assigned with labels which summarize their contents, labeling fake news can be time-consuming, and labeling drugs according to their toxicity requires expensive wet lab experiments or analysis [29, 95, 96, 20]. *Contrastive learning* (CL) is an increasingly popular unsupervised graph representation learning paradigm for such label scarce settings [1, 97, 51, 14, 98] and is currently the state-of-the-art in unsupervised visual representation learning [54, 55, 56, 59].

Broadly, CL frameworks learn representations by maximizing similarity between augmentations of a sample (positive views) while simultaneously minimizing similarity to other samples in the batch (negative views). Recent theoretical and empirical works attribute the impressive success of visual CL (VCL) to two key principles: (i) leveraging *strong, task-relevant data augmentation* [99, 100, 101, 102, 103] and (ii) training on *large, diverse datasets* [54, 55, 104, 105, 106]. By using appropriate data augmentations, VCL frameworks learn high quality representations that are *invariant* to properties *irrelevant* to downstream task performance; thereby preserving task-relevant properties and preventing the model

The material in this chapter is derived from the paper "Augmentations in Graph Contrastive Learning: Current Methodological Flaws & Towards Better Practices" [79], which appeared in the proceedings of the ACM Web Conference 2022. Code can be accessed here.

from learning brittle shortcuts [99, 102, 54, 107]. Large, diverse datasets are necessary as VCL frameworks routinely use 1K–8K samples in a batch to ensure that enough negative views are available to train stably [54, 55, 56, 108]. Representations learnt using VCL and self-supervised learning in general have been found to be more robust [57], transferable [109] and semantically aligned [110] than their supervised counterparts.

Interestingly, graph CL (GCL) frameworks often deviate from these key principles and yet report seemingly strong task performance. Small, binary graph classification datasets [111] are routinely used to benchmark GCL frameworks. Moreover, due to the non-euclidean, discrete nature of graphs, it can be difficult to design task-relevant graph data augmentations [112, 113] or know what invariances are useful for the downstream task. Therefore, frameworks often rely upon domain-agnostic graph augmentations (DAGAs) [1]. However, DAGAs can destroy task relevant information and yield *invalid/false positive* samples (see Fig. 3.1). It is also unclear if DAGAs induce invariances that are useful or semantically meaningful with respect to the downstream task.

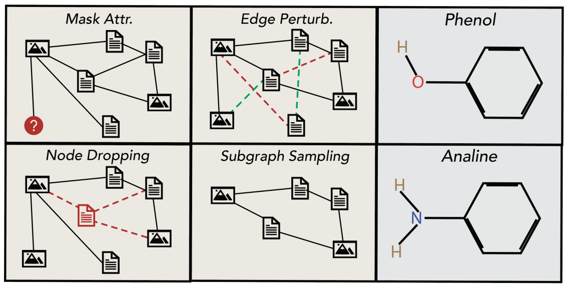


Figure 3.1: **Domain-Agnostic Graph Augmentations (DAGAs).** [Left] introduced in [1]. Deletion/addition in red/-green. **False Positive Samples.** [Right] Acidic molecule Phenol and basic molecule Aniline are structurally similar but have different properties. DAGAs can inadvertently generate this pair as a positive view, resulting in *similar* representations for semantically *dissimilar* entities.

In this work, we investigate the implications of the aforementioned discrepancies by probing the quality of representations learnt by popular GCL frameworks using DAGAs. We show that DAGAs can destroy task-relevant information and lead to weakly discriminative representations. Moreover, on popular, small benchmark datasets, we find that flawed evaluation protocols and the strong inductive bias of GNNs mitigate limitations of DAGAs. Our analysis offers several actionable sanity checks and better practices for practitioners when evaluating GCL representation quality. Further, through two case studies on larger, more complex datasets, we demonstrate that task-aware augmentations (TAAs) are necessary for strong performance and discuss how to identify such augmentations amenable to GCL. Our main contributions are summarized as follows:

- **Analysis of limitations in domain-agnostic augmentations:** We demonstrate that commonly-used DAGAs lead models to learn weakly discriminative representations by inducing invariances to invalid views or false-positives. Across several architectures and datasets, we find these shortcomings are mitigated by the strong inductive bias of GNNs, which allow existing methods to achieve competitive results on benchmark datasets.

- **Identification of methodological flaws & better practices**: We contextualize recent theoretical work in visual self-supervised learning to identify problematic practices in GCL: (i) the use of small datasets and (ii) training with negative-sample frameworks on binary classification datasets. Furthermore, we provide carefully-designed sanity checks for practitioners to assess the benefits of proposed augmentations and frameworks.

- **Case studies with strong augmentations:** In two case studies on different data modalities, we demonstrate how to leverage simple domain knowledge to develop strong, task-aware graph augmentations. Our systematic process results in up to 20% accuracy improvements.

3.2 Background & Related Work

We begin by introducing CL. We then discuss how strong, task-relevant augmentations and large, diverse datasets underpin the success of VCL. Finally, GCL and graph data augmentation are discussed. Please see section A.4 for additional related work.

3.2.1 Contrastive Learning (CL)

Frameworks & Losses. Several CL frameworks [54, 55, 7] have been proposed to enforce similarity between positive samples and dissimilarity between negative samples, where positive samples are generated through data augmentation. Normalized temperature-scaled cross entropy (NT-XENT) is a popular objective used by several state-of-the-art CL frameworks [54, 114, 115, 116, 97, 1, 117] and is defined as follows. Let $\mathcal{X}$ be a data domain, $\mathcal{D} = \{x_{[1\dots n]} | x_i \in \mathcal{X}\}$ be a dataset, $\mathcal{T}\colon \mathcal{X} \to \tilde{\mathcal{X}}$ be a stochastic data transformation that returns a positive view, and $f\colon \{\mathcal{X}, \tilde{\mathcal{X}}\} \to \mathbb{R}^d$ be an encoder. Further, assume we are given a batch of size N, similarity function $\mathrm{sim}\colon (\mathbb{R}^d, \mathbb{R}^d) \to [0,1]$, temperature parameter τ, and encoded positive pair $\{\boldsymbol{z}_i, \boldsymbol{z}_j\}$. Then, NT-XENT can be defined as:

$$\ell_{i,j} = -\log \frac{\exp\left(\mathrm{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp\left(\mathrm{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_k\right)/\tau\right)}. \tag{3.1}$$

Here, the numerator encourages the positive pair to be similar, while the denominator encourages negative pairs ($k \neq i$) to be dissimilar. Alternative CL objectives may enforce such (dis)similarity differently (e.g., through margin maximization [118] or cosine similarity

[7]), but the principles discussed below uniformly explain the success of contrastive learning frameworks [119].

The role of augmentations. Recent work [99, 120, 102] has demonstrated that data augmentation is critical for training CL frameworks. Theoretically, Tian et. al [99] show that positive views should preserve task-relevant information, while simultaneously minimizing task-irrelevant information [120]. Training on such views introduces invariances to irrelevant information, leading to more generalizable representations. Indeed, state-of-the-art VCL frameworks [54, 55, 56, 121, 122] rely upon strong, task relevant data augmentation to generate such views. For example, Purushwalkam et al. [102] show that augmentations used by SimCLR introduce "occlusion invariance", which is useful in classification tasks where objects may be occluded. Overall, we highlight that augmentation strategies are not universal [1, 97] and must align with the task; e.g., semantic segmentation tasks would benefit more from augmentations that induce view-point invariances [102].

The role of large, high-quality datasets. Empirically, CL frameworks [54, 55, 108] often require many negative samples in each batch to avoid class collisions (i.e., false positives) [119]. Further, recent theoretical work has shown that optimizing Eq. (3.1) is equivalent to learning an estimator for the mutual information shared between positive views, where the quality of this estimate is upper-bounded by batch-size [114, 104]. These properties combine to necessitate the use of large, diverse datasets in contrastive learning.

3.2.2 Graph Contrastive Learning (GCL)

Frameworks. In this paper, we focus on three state-of-the-art unsupervised representation learning frameworks for graph classification that represent different methodological perspectives: GraphCL [1], InfoGraph [14] and MVGRL [51]. Similar to SimCLR, GraphCL uses NT-XENT to contrast representations of augmented samples using a shared encoder. Much like DeepInfoMax [123], InfoGraph maximizes the mutual information between local and global views, where corresponding views are obtained through subgraph sampling and graph-pooling. Meanwhile, MVGRL mirrors CMC [124] and uses dual encoders to contrast multiple views of a graph, where views are generated by first running a diffusion process (e.g. Personalized Page Rank [125], Heat Kernel [126]) over the graph and then sampling subgraphs.

Graph data augmentation. Existing GCL frameworks leverage three main strategies to generate views: feature or topological perturbation (GraphCL), sampling (InfoGraph), and/or diffusion processes (MVGRL). We focus on the domain-agnostic graph augmentations (DAGAs) introduced by GraphCL, shown in Fig. 3.1, as these are more popular in recent frameworks [97, 52, 1], composable [54, 122], fast, and do not require dual view encoders. An empirical study on the benefits of DAGAs in GCL [1] demonstrates that (i) composing augmentations and adjusting augmentation strength to create a more difficult instance discrimination task improves downstream performance and (ii) augmentation utility is dataset dependent. However, a critical assumption underlying DAGA is that by limiting augmentation strength such that only a fraction of the original graph is modified, task-relevant information is not significantly altered. In Sec. 3.3, we revisit this assumption to show that it does not hold for many datasets and discuss the implications of training with poorly augmented graphs. Clearly, it is expected that models trained with task-aware augmentations (TAAs) that induce useful invariances will learn better features than those trained with DAGAs. However, graphs are often used as abstracted representations of structured data, such as molecules [20] or point clouds [127], and it is often unclear how to represent task-relevant invariances after abstracting to the graph space. In Sec. 3.4, we discuss a broad strategy for identifying augmentations that induce task-relevant invariances in the abstracted, graph space and demonstrate the significant performance boosts achieved by using such augmentations.

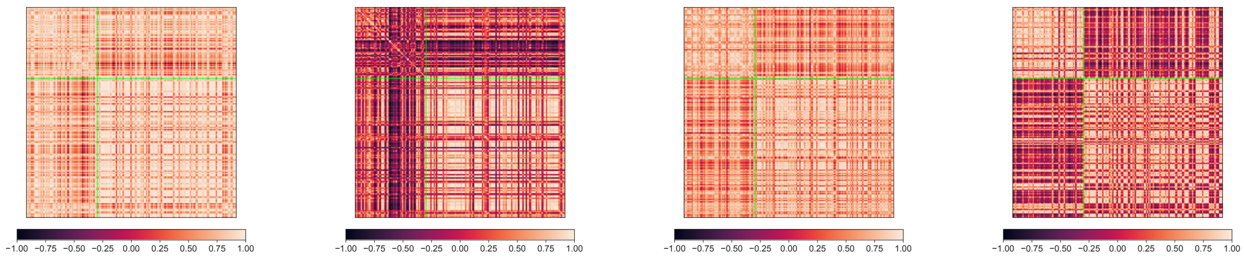
Automated Graph Data Augmentation. Concurrent works [117, 97, 128, 129, 130, 131, 132, 133] have begun investigating *automated* graph data augmentation as a means of both avoiding costly trial and error when selecting augmentations and generating more informative, task relevant views. These methods often use bi-level optimization objectives and/or viewmakers [134] to jointly learn representations and augmentations (cf. Appendix A.4 for more details). Our analysis (Sec. 3.3) remains pertinent for GCL with automated augmentations. Namely, the proposed sanity checks are not augmentation specific, the identified evaluation flaws must still be considered, and untrained models should still be included as baselines. Also, our discussion on the benefits and properties of TAAs (Sec. 3.4) remains relevant as it is difficult to identify post-hoc if an automated augmentation strategy is inducing semantically meaningful invariances or exploiting shortcuts.

3.3 Revisiting Augmentations & Evaluation in GCL

In this section, we investigate how existing GCL frameworks deviate from the principles underlying the success of VCL methods and the effects of such deviations. We discuss and establish three key observations:

(O1) Standard graph data augmentation is susceptible to altering graphs semantics and task-relevant information.

(O2) Training on such augmentations can lead to weakly discriminative representations.



(a) Random Init. (85.76 ± 7.38)

(b) GraphCL (86.80 ± 1.34)

(c) InfoGraph (89.01 ± 1.13)

(d) MVGRL (89.70 ± 1.1)

Figure 3.2: **Representational Similarity.** The normalized cosine similarity between all-pairs of representations is shown above for the MUTAG dataset. The on-diagonal blocks (indicated by green lines) show intra-class similarity, while off-diagonal blocks show inter-class similarity. MVGRL, which uses diffusion-based views, learns representations that have high *intra*-class similarity and low *inter*-class similarity, as desired. InfoGraph, which directly maximizes mutual information between local/global views, preserves high intra-class similarity, and has moderate inter-class similarity. GraphCL, which uses domain-agnostic graph augmentations, has low intra-class similarity in the upper left block. This indicates that training on false positive/invalid samples can negatively impact representational power.

(O3) The strong inductive bias of randomly-initialized GNNs obfuscates the performance of weak representations and misaligned evaluation practices.

Empirical Setup. In our analysis, we focus on commonly used graph classification datasets (Table 3.1) [111]. Official implementations for GraphCL*, InfoGraph†, and MVGRL‡ are used. We consider the encoder architecture used by [1] and report results with graph convolutional layers from GIN [27] (original implementation), PNA [26], SAGE [28], GAT [25], and GCN [24]. See section A.1 for details on the training setup.

3.3.1 (O1) Domain-agnostic graph augmentations alter task-relevant information

Given the importance of data augmentation in representation learning, several works [135, 99, 102, 100, 101] have investigated its properties. Recently, Gontijo-Lopes et al. [13] identified an empirical trade-off when selecting amongst augmentations to improve model generalization. Intuitively, augmentations should generate samples that are close enough to the original data to share task-relevant semantics and different enough to prevent trivially similar samples.

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

†https://github.com/fanyun-sun/InfoGraph

‡https://github.com/kavehhassani/mvgrl

Table 3.1: **Dataset Description**

Name	**Graphs**	**Classes**	**Avg. Nodes**	**Avg. Edges**	**Domain**
IMDB-BINARY [136]	1000	2	19.77	96.53	Social
REDDIT-BINARY [136]	2000	2	429.63	497.75	Social
GOSSIPCOP [137]	5464	2	55.48	54.51	News
DEEZER [91]	9629	2	23.49	65.25	Social
GITHUB SGZR [91]	12725	2	113.79	234.64	Social
MUTAG [138]	188	2	17.93	19.79	Molecule
PROTEINS [139]	1113	2	39.06	72.82	Bioinf.
DD [140]	1178	2	284.32	715.66	Bioinf.
NCI1 [93]	4110	2	29.87	32.30	Molecule

This trade-off can be quantified through two metrics, *affinity* and *diversity*. Affinity measures the distribution shift between the augmented and original sample distributions. Diversity quantifies how difficult it is to learn from augmented samples instead of only training samples [13]. While augmentations that best improve generalization optimize for both metrics [13], it is not clear that DAGAs also optimize for both. For example, molecular graph classification tasks are commonly used to evaluate GCL frameworks. However, as noted in Fig. 3.1, limited perturbations are needed to invalidate a molecule or significantly alter its function. Here, augmented data is sufficiently diverse, but it is not clear if creating invalid molecule samples also leads to low affinity, indicating that task-relevant information have been destroyed. We conduct the following experiment to understand the affinity of DAGAs on benchmark datasets.

Experimental setup. We measure affinity as follows: (i) train a supervised PNA encoder on the original training data, (ii) generate an augmented dataset by using random node/subgraph dropping at 20% of the graph size, as suggested by [1] and (iii) evaluate on clean *and* augmented training data separately. The difference between clean and augmented accuracy quantifies the distribution shift induced by augmentations [13].

Hypothesis. We argue that while it is not expected that accuracy on augmented data will match that of clean data, augmented accuracy should be nontrivial if augmentations are indeed information-preserving [100, 103].

Results. In Table 3.2, we see a considerable difference between clean and augmented accuracy across datasets. This implies low affinity, i.e., a large shift between augmented and training distributions, and confirms that DAGAs can destroy task-relevant information. Consequently, training on such samples will harm downstream task performance, as shown by prior works on VCL [103] and elucidated below for GCL.

Table 3.2: **Augmentation Affinity**. Affinity [13], measured by the difference between original and augmented accuracy of a supervised model, captures how much the data distribution has changed as a result of augmentation. We see that DAGAs lead to low affinity. This is expected for molecular datasets, where it is easy to create invalid molecules, and is also true for some social network datasets.

Dataset	**Clean Train Acc.**	**Aug. Train Acc.**
MUTAG	90.14 ± 1.36	37.67 ± 1.48
PROTEINS	70.70 ± 4.30	56.54 ± 8.11
NCI1	75.55 ± 4.60	60.15 ± 0.069
DD	84.06 ± 8.81	65.41 ± 14.87
REDDIT-BINARY	85.56 ± 3.21	50.56 ± 0.09
IMDB-BINARY	70.93 ± 0.046	50.11 ± 0.384
GOSSIPCOP	98.047 ± 0.37	96.03 ± 1.57

3.3.2 (O2) Domain-agnostic augmentations induce weak discriminability

Recall that contrastive losses maximize the similarity between representations of positive pairs while simultaneously minimizing the similarity amongst representations of negative samples. However, Obs. (**O1**) identifies that DAGAs have low affinity, which suggests that task-relevant information has been significantly altered. This implies that representation similarity will be maximized for samples that are *not* semantically similar, e.g., false positive samples. Consequently, the resulting representations may not be discriminative with respect to downstream classes—i.e., *intra*-class samples may have lower similarity than *inter*-class samples, counter to what is expected. This claim is investigated in the following experiment.

Experimental setup. We measure the discriminative power of representations learned using GCL as follows: given models trained using GraphCL, InfoGraph and MVGRL, we extract representations for the entire dataset. Then, we calculate cosine similarity between all representation pairs. Representational similarity from an untrained model is also included.

Hypothesis. If a model has learned discriminative representations, intra-class similarity should be high while inter-class similarity should be low.

Results. In Fig. 3.2, we plot the normalized cosine similarity between representations (sorted by class label), such that the upper left and lower right quadrants correspond to the similarity between same-class representations. Results on additional datasets can be found in Appendix A.1. We see that MVGRL (Fig. 3.2d) and InfoGraph (Fig. 3.2c) are less likely to encounter false positive pairs as they, respectively, use diffusion-based views and maximize mutual information over sampled subgraphs. GraphCL, which uses DAGAs, is more

Table 3.3: **Inductive Bias on Benchmark Datasets.** Following the same evaluation protocol as [14], we generate embeddings from an untrained N-Layer GIN encoder and perform classification using an SVM classifier. Results for GraphCL and InfoGraph are reported from [1]. Best accuracy is in bold; other models whose accuracy with standard deviation falls within the standard deviation of the best accuracy are underlined. We see across all datasets that untrained models have a strong inductive bias. On PROTEINS, DD, MUTAG DEEZER and GITHUB-SGZR, untrained models perform competitively against trained models.

Dataset (# Samples)	**Random Init** (3 layers)	**Random Init** (4 layers)	**Random Init** (5 layers)	**GraphCL** [1]	**InfoGraph** [14]
IMDB-BINARY (1000)	67.22 ± 7.77	61.26 ± 7.01	60.43 ± 5.92	71.14 ± 0.44	$\mathbf{73.03 \pm 0.87}$
REDDIT-BINARY (2000)	72.34 ± 6.64	64.57 ± 8.03	67.32 ± 7.41	$\mathbf{89.53 \pm 0.84}$	82.50 ± 1.42
DEEZER (9629)	$\mathbf{56.59 \pm 0.01}$	54.99 ± 1.74	54.87 ± 2.60	56.19 ± 0.015	55.89 ± 0.88
GITHUB SGZR (12725)	64.51 ± 0.05	64.93 ± 0.04	64.93 ± 0.89	$\mathbf{65.81 \pm 0.413}$	Out of Time
MUTAG (188)	85.76 ± 7.38	86.36 ± 6.51	86.73 ± 10.33	86.80 ± 1.34	$\mathbf{89.01 \pm 1.13}$
PROTEINS (1113)	73.64 ± 5.464	$\mathbf{74.46 \pm 4.09}$	74.22 ± 2.85	74.39 ± 0.45	74.44 ± 0.31
DD (1178)	73.23 ± 8.25	72.15 ± 7.25	77.08 ± 4.18	$\mathbf{78.62 \pm 0.40}$	72.85 ± 1.78
NCI1 (4110)	70.65 ± 1.99	70.36 ± 3.11	70.49 ± 2.42	$\mathbf{77.81 \pm 0.41}$	76.20 ± 1.06

likely to encounter false positive samples that can harm discriminative power (Obs. (**O1**)). Correspondingly, MVGRL and InfoGraph both learn representations with higher intra-class similarity than inter-class similarity. In contrast, GraphCL has low intra-class similarity as can be seen in the upper-left quadrant (Fig. 3.2b). This implies that the model has not learned features that capture the semantic similarity between the samples belonging to this class. However, we note that while MVGRL has learned discriminative representations, it requires dual encoders and it is unclear what invariances are learnt by training with diffusion-based views. Finally, we find that even though the randomly initialized, untrained model (Fig. 3.2a) has higher absolute values for average intra- and inter-class similarities than trained methods, it achieves inter-class similarity relatively lower than intra-class similarity, as required for discriminative applications. We further elaborate on this point in the next section.

Proposed evaluation practice. Given that CL frameworks directly optimize the similarity between representations, we argue that plotting representational similarity can serve as a simple sanity check for practitioners to assess the quality of their model's learned representations. Indeed, models are often only assessed through linear evaluation or task accuracy, which may hide differences in the discriminative power of representations. For example, as shown in Fig. 3.2, InfoGraph and MVGRL have similar task accuracy, but MVGRL has learnt more discriminative representations.

Having established that DAGAs can lead to invalid or false positive augmented samples and that training on such samples can lead to poorly-discriminative representations, we next investigate whether other factors are bolstering GCL performance. Specifically, we discuss the role of randomly initialized, untrained GNN inductive bias and identify flaws in current GCL evaluation practices.

3.3.3 (O3) Strong inductive bias of random models reduces GCL inefficiencies

As noted in Obs. (**O2**), randomly-initialized, untrained GNNs can produce representations that are already discriminative without any training (Fig. 3.2a). While the strength of inductive bias of GNNs in (semi-) supervised settings has been noted before [24, 141, 142, 117], we aim to better contextualize the performance of GCL frameworks by conducting a systematic analysis of the inductive bias of GNNs, using several datasets and architectures. Understanding the performance of untrained models helps contextualize the cost of training.

Empirical setup. For DEEZER and GITHUB-SGZR, a PNA encoder is used to stabilize training. All other datasets are trained with a GIN encoder. MVGRL ran out-of-memory so we did not include it in this evaluation. See Appendix A.1 for more details.

Results. As shown in Table 3.3, randomly-initialized, untrained models perform competitively against trained models on several benchmark datasets. It is likely that some of the negative effects of training with DAGAs (Obs. (**O1**)–(**O2**)) were mitigated by this strong inductive bias. However, note that it becomes difficult to justify the additional cost of GCL on datasets where task performance and representation quality are not noticeably better than untrained models. Below, we discuss how to fairly evaluate GCL frameworks and how popular benchmark datasets are, in fact, inappropriate for GCL.

Proposed evaluation practices. Given that randomly-initialized, untrained models are a non-trivial baseline for GCL frameworks, we argue that they should be included when evaluating novel frameworks to contextualize the benefits of unsupervised training. While some recent works [143, 117] include untrained models in their evaluation, this practice remains far from standardized.

Furthermore, CL frameworks often define negative samples through the other samples in the batch. Given the limited size of popular benchmark datasets (Table 3.3), it can be difficult to ensure that each batch is large enough to train stably. Further, given that these benchmarks are often binary classification tasks, half the samples, in a balanced setting, are expected to share the positive pair's label but be treated as negative samples. This implies that representations learned with GCL may not be discriminative because models have minimized similarity for semantically related examples. We thus argue that evaluating *GCL* frameworks on these datasets is flawed and this practice should be discontinued.

We highlight that Dwivedi et al. [86] also find popular graph classification datasets are

Table 3.4: **Document Classification.** We use domain-agnostic subgraph dropping (S) and node-dropping (N) at 10% and 5% of sentence length, respectively, for baseline augmentations. For task-aware augmentations, we stochastically apply synonym replacement (5%), random insertion (5%), random swapping (5 %) and random deletion (10 %). Random Accuracy with window-size = 2 is 58.46 ± 1.97. Random Accuracy with window-size = 4 is 63.93 ± 0.045.

	GCN		**SAGE**		**GIN**	
Augmentation	**SimSiam Acc.**	**BYOL Acc.**	**SimSiam Acc.**	**BYOL Acc.**	**SimSiam Acc.**	**BYOL Acc.**
S. vs. S (ws =2)	69.41 ± 7.28	62.98 ± 3.12	59.17 ± 8.36	67.17 ± 2.70	55.67 ± 4.61	65.02 ± 2.00
S vs. N (ws =2)	57.84 ± 4.31	65.78 ± 8.22	56.74 ± 1.70	63.77 ± 2.90	58.2 ± 8.24	74.26 ± 3.80
Context-Aware (ws = 2)	$\mathbf{83.65 \pm 2.31}$	$\mathbf{78.12 \pm 2.73}$	$\mathbf{81.28 \pm 2.54}$	$\mathbf{78.23 \pm 4.53}$	$\mathbf{80.37 \pm 4.07}$	$\mathbf{77.79 \pm 0.09}$
S vs. S (ws = 4)	61.76 ± 5.12	66.38 ± 2.29	54.68 ± 1.53	67.37 ± 1.11	54.71 ± 3.00	66.18 ± 2.34
S vs. N (ws = 4)	55.38 ± 1.99	68.311.88	59.23 ± 8.03	70.6 ± 4.85	53.31 ± 1.36	66.59 ± 1.57
Context-Aware (ws = 4)	$\mathbf{81.12 \pm 3.97}$	$\mathbf{74.05 \pm 5.465}$	$\mathbf{80.67 \pm 10.36}$	$\mathbf{75.65 \pm 5.54}$	$\mathbf{75.30 \pm 15.61}$	$\mathbf{76.55 \pm 7.43}$

problematic in general, but for the specific case of GraphCL, this point is of some urgency as such small-scale datasets are part of standard GCL evaluation [97, 117]. However, we note that self-supervised frameworks that do not rely on negative samples, such as BYOL[122] and SimSiam[121], can be used as an appropriate alternative for binary datasets. Such frameworks maximize similarity between sample augmentations and avoid degenerate solutions via stop-gradient operations and exponentially moving average target networks.

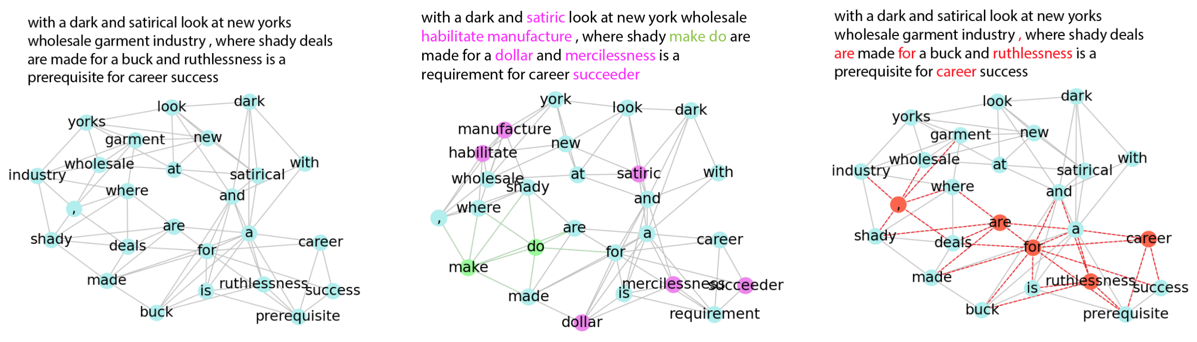
3.3.4 Summary of Proposed Evaluation Practices

We summarize the practices that we hope will be adopted in future graph CL research:

- Given that DAGA can destroy task-relevant information and harm the model's ability to learn discriminative representations, there is need for designing context-aware graph augmentations (Sec. 3.4).
- Randomly initialized, untrained GNNs have strong inductive bias and should be reported during evaluation.
- Small, binary graph datasets are inappropriate for evaluating GCL frameworks.
- GCL frameworks should be comprehensively evaluated using metrics beyond accuracy to assess representation quality.

3.4 Benefits & Design of Task-Aware Augmentations

In this section, we exemplify the benefits of adhering to key VCL principles by defining a broad strategy for finding task-aware augmentations in scenarios where prior domain knowledge is


with a dark and satirical look at new yorks wholesale garment industry , where shady deals are made for a buck and ruthlessness is a prerequisite for career success

with a dark and satiric look at new york wholesale habilitate manufacture , where shady make do are made for a dollar and mercilessness is a requirement for career succeeder

with a dark and satirical look at new yorks wholesale garment industry , where shady deals are made for a buck and ruthlessness is a prerequisite for career success

(a) Original (b) Natural Language Space (c) Graph Space

Figure 3.3: **Augmentations for Document Classification.** Documents are represented as co-occurrence graphs [2, 3], where words are treated as nodes with word2vec embeddings, and edges indicate co-occurrence in a sliding windows [4]. As shown in (b), we perform synonym replacement (purple) and random word insertion (green) to augment sentences without losing task-relevant information [5]. In (c), we show random node (word) deletion (red). Our results show that natural language space augmentations improve classification accuracy substantially over baseline augmentations.

available. We note the goal of this strategy is not to resolve problematic data augmentations in GCL. Instead, we use the proposed strategy to help elucidate the benefits of abiding by VCL principles in two careful case studies.

Augmentation strategy: For many graph-based representation learning tasks, structured data, such as documents [2], propagation patterns [83], molecules [95], maps [144], and point-clouds [127], are first abstracted as graphs via a deterministic process before task-specific learning can begin. In this practical setting, our idea is to leverage knowledge pertaining to the original, structured data to find augmentations that will, in the abstracted graph space, (i) preserve task-relevant information, (ii) break view symmetry, and (iii) introduce semantically meaningful invariance. In our first case study, which focuses on a graph-based document classification task, we achieve this goal by exploiting existing natural language augmentations [5] and directly perturbing the raw input before its graph is constructed. However, when given a sufficiently complex graph construction process, it can be unclear if augmentations in the original space will induce useful invariances or retain task-relevant information in the abstracted graph space. In our second case study, which focuses on image classification using super-pixel nearest-neighbor graphs, we encounter this setting and propose to avoid destruction of task-relevant information by deliberately introducing task-*irrelevant* information. We then use augmentations designed to induce invariance to such *irrelevant* information.

3.4.1 Case Study 1: Document Classification

We first focus on a binary graph-based document classification task. As shown by prior work [89], graph-based representations are effective at capturing not only the content but also the structure of a document, leading to improved classification performance in this setting. Here, our goal is to demonstrate adhering to VCL principles by using TAAs is needed to improve task performance.

Dataset & Task. The task is to classify movie reviews and plot summaries according to their subjectivity. Following [2], we convert the Subjectivity document dataset [145] (10k samples) into co-occurrence graphs, where nodes represent words, edges indicate that two words have co-occurred within the same window (e.g. window size 2 and 4), and node features are word2vec [4] embeddings. An example of this conversion is shown in Fig. 3.3a. Note that we only use positive-view-based self-supervised learning frameworks (e.g., SimSiam, BYOL) because this is a binary classification task (see Sec 3.3.3). Accuracy is computed using a kNN classifier.

Setup of GNN models. We use a Message Passing Attention Network [2] as the encoder, and a 2-layer MLP as the predictor. The representation dimension is 64, and models are trained using Adam [146] with LR=5e-4. Additional training details are given in Appendix A.2. We report results with the original GCN layer used by [2], as well as with GraphSAGE [28] and GIN [27] layers replacing it.

Domain-Agnostic Graph Augmentations. We conduct an informal grid search to select which DAGAs and augmentation strengths to use. Among node, edge, and subgraph dropping at $\{5\%, 10\%, 20\%\}$ of text length, we find generating both views using subgraph dropping (10%) performs the best. Generating one view with subgraph dropping (10%) and the other with node-dropping (10%) performs second best. We evaluate both strategies.

Task-Aware Augmentations. Recently, Wei et al. [5] proposed several intuitive augmentations for use in natural language processing, namely: synonym replacement, random word insertion, random word swapping and random word deletion, where the augmentation strength is determined by the sentence length. (See Fig. 3.3b for an example.) By design, these augmentations introduce invariances that are useful to downstream tasks (e.g., invariance to the occasional dropped word), preserve task-relevant information, and break view symmetry in the natural language modality. Due to a co-occurrence based construction process, changes in the underlying document will manifest in the corresponding graph, so it is likely that augmentations remain effective for the abstracted space.

Results. As shown in Table 3.4, task-relevant, natural language augmentations perform considerably better (up to +20%) than domain agnostic graph augmentations for both window sizes. Notably, TAAs are necessary to significantly improve performance over an untrained baseline, indicating that adhering to key principles of VCL is indeed beneficial.

Potential Graph Space Augmentations. While natural language augmentations modify samples prior to the graph construction process, it is easy to see that they can be converted into graph augmentations, effectively infusing DAGAs with domain knowledge on how to perturb co-occurence graphs. Specifically, synonym replacement is equivalent to replacing node features of the selected word (node) with the closest word2vec embedding. Random insertion can be approximated in the co-occurence graph by (i) creating a new node with a randomly selected word2vec embedding and (ii) duplicating the connections of an existing node. Random deletion can be represented by (i) randomly removing a node and (ii) rewiring the modified graph to connect neighbors of the removed node. Random swap is equivalent to swapping the features of two nodes. We highlight that domain-agnostic subgraph and node dropping do *not* rewire the co-occurence graph. Thus, it is unclear what invariance to these augmentations represents in the original data modality. In Appendix A.2, we show that graph-space and document-space synonym replacement perform comparably, but leave the evaluation of other converted graph space augmentations to future work.

In this case study, we were able directly leverage augmentations in the original modality, which are known to preserve task-relevant information and induce useful invariances, to significantly outperform DAGAs. The next case study focuses on a more challenging setting where augmentations in the original modality are not immediately amenable to GCL due to a complex graph construction process and GNN architectural invariances.

3.4.2 Case Study 2: Super-pixel Classification

Our second case study is based on super-pixel MNIST classification, a standard benchmark for evaluating GNN performance [86, 147]. Here, we pursue an alternative strategy for task-aware augmentation where augmentations must induce invariance to deliberately irrelevant information (e.g., color for digit classification).

Dataset & Task. We follow the established protocols in [147, 86] to create super-pixel representations of MNIST, where each image is represented as a k-nearest neighbors graph between super-pixels (homogeneous patches of intensity). Nodes map to super-pixels, node features are super-pixel intensity and position, and edges are connections to k neighbors. An example is shown in Fig. 3.4.

Setup of GNN models. The following architecture is used for experiments. The encoder is 5-layer GIN architecture similar to [1] and [86]. The projector is a 2-layer MLP and there is no predictor. Models are trained for 80 epochs, using Adam [146] with LR of 1e-3, and

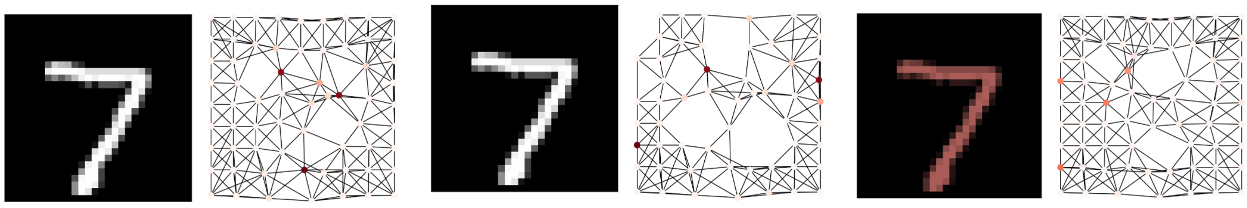

(a) Original (b) Node Dropping (c) Colorizing

Figure 3.4: **Augmentations for Super-pixel Classification.** Node dropping alters graph topology and it is unclear if task-relevant information is preserved. Colorizing preserves task-relevant information by only perturbing node features.

Table 3.5: **Super-pixel Classification.** KNN Accuracy after unsupervised training with Node Dropping or context aware graph augmentations (Colorize) is reported. Context aware augmentations improve performance. Accuracy of randomly initialized model is 37.79 ± 0.03.

Aug.	**SimSiam Acc.**	**SimCLR Acc.**	**BYOL Acc.**
Node Dropping (20%)	66.30 ± 0.33	68.56 ± 0.16	$\mathbf{65.32 \pm 0.95}$
Node Dropping (30%)	61.30 ± 0.48	68.07 ± 0.37	61.87 ± 1.03
Colorize	$\mathbf{68.95 \pm 1.20}$	$\mathbf{73.67 \pm 0.10}$	$\underline{64.42 \pm 2.385}$

the representation dimension is set to 110. The models are trained using SimSiam [121], BYOL [122], and SimCLR [54]. We give more training details in section A.3. While composing augmentations is known to improve performance on vision tasks, we avoid it here in order to fairly compare to graph baselines, which only consider a single augmentation.

Domain-Agnostic Graph Augmentations. Following [1], we apply random node dropping at 20% of the graph size to obtain both samples in the positive pair.

Task-Aware Augmentations. While geometric image augmentations [54], such as horizontal flipping and rotating, generally preserve task-relevant information and introduce semantically meaningful invariance, they cannot break view symmetry in GCL frameworks as GNNs are permutation invariant. Therefore, the representations of a pair of flipped images will be similar as their corresponding super-pixel graph representations are equivalent up to node reordering. On the other hand, augmentations such as cropping may result in qualitatively different super-pixel graphs. Here, it is unclear if the super-pixel graph obtained after augmentation preserves task-relevant information, even if cropping is information preserving with respect to the original image. Therefore, it is not trivial to identify successful augmentations in the abstracted domain that will also be successful in graph space.

Given the difficulty of identifying augmentations that perturb super-pixel graph topology but also preserve task-relevant information, we focus on image space augmentations that lead to modified node features in the super-pixel graph. Specifically, we select *random colorization* as the TAA as it (i) preserves task-relevant information as color is not relevant property when classifying digits, (ii) breaks view symmetry because the node features of augmented samples are different and (iii) introduces a harmless invariance to colorization. We briefly note that augmentations are generally selected to introduce invariances that are useful to the downstream task. For example, cropping results in occlusion invariance, which is useful for classification tasks where objects are often partially covered [102]. Here, we take a complementary approach where augmentations introduce harmless information (color) and the model learns to ignore it. This can be a useful strategy when it is difficult to clearly identify potentially useful invariances for a given task.

Results. In Table 3.5, we observe that training with an information-preserving, TAA (colorizing) improves accuracy for both SimSiam and SimCLR. While BYOL generally performs worse than SimSiam and SimCLR, colorizing is still within standard deviation of DAGAs. Composing augmentations with colorizing would likely further improve performance, but this investigation is left to future work. This confirms that learning invariance to irrelevant information, as determined by knowledge of the original data modality, is indeed a viable strategy for creating TAAs. Moreover, we note that randomly-initialized models have 37.79% accuracy, indicating that super-pixel data can serve as a sufficiently complex benchmark for future GCL evaluation [86] (see Appendix A.3 for affinity and representational similarity analysis).

3.5 Conclusion

In this work, we discuss limitations in the evaluation and design of existing instance-discrimination GCL frameworks, and introduce new improved practices. In two case studies, we show the benefits of adhering to these practices, particularly the benefits using task-aware augmentations. First, through our analysis, we show that domain-agnostic graph augmentations do not preserve task-relevant information and lead to weakly discriminative representations. We then demonstrate that benchmark graph classification datasets are not appropriate for evaluating GCL frameworks by contextualizing recent theoretical work in VCL. Indeed, we show that the strong inductive bias of randomly initialized, untrained GNNs obfuscates GCL framework inefficiencies. While we acknowledge the community is moving toward larger and more extensive benchmarks [86], we emphasize that it is fundamentally incorrect to continue evaluating GCL on legacy graph classification benchmarks. Furthermore, on two case studies with practically complex tasks, we show how to use domain knowledge to perform information-preserving, task-aware augmentation and achieve significant improvements over training with domain-agnostic graph augmentations. In summary, GCL is an exciting new direction in unsupervised graph representation learning and our work can inform the evaluation of new methods as well as help practitioners design task-aware augmentations.

CHAPTER 4

Analyzing Data Centric Properties for Graph Contrastive Learning

4.1 Introduction

In the preceding chapter, we empirically identified several limitations in popular augmentation strategies and proposed better practices. In this chapter, we take a complementary lens that allows us to formally analyze properties of augmentations through a rigorous generalization analysis and flexible synthetic data generating process. Together, these chapters allow Part I to provide considerable insights into learning more powerful graph representations and accessing the benefits beyond accuracy, e.g., robustness [57, 58], transferability [60, 95], and semantic consistency [59], that self-supervised learning (SSL) [54, 121, 148, 55, 56, 59, 124, 114, 123] has driven in visual representation learning

Ineed, this impressive empirical success has motivated a surge of efforts that seek to gain insights into SSL's behavior [119, 7, 99, 100, 101, 8, 102, 122] or adapt successful frameworks to different modalities, including graph data [1, 51, 52, 149, 14]. Notably, many analyses of SSL have converged upon the following data-centric properties as critical to its success: (i) augmentations should induce *invariance* to task-irrelevant attributes, as to better reflect the underlying data generation process and improve generalizability; (ii) samples (and corresponding augmentations) from different underlying classes should be *separable* in some latent space, as to ensure a high-performing classifier is realizable; and (iii) labels of augmented samples should be *recoverable* from the natural sample using which they were generated [99, 102, 120] so that representations are semantically consistent for downstream

The material in this chapter is derived from the paper "Analyzing Data-Centric Properties for Graph Contrastive Learning" [80], which appeared in the proceedings of NeurIPS 2022. Code can be found here.

tasks. Due to the continuous representation of natural images and well-designed augmentation strategies, these properties are indeed aligned with standard visual SSL practices [150].

However, despite the growing popularity of SSL for graph representation learning, it appears unlikely that the above properties are supported for non-Euclidean, discrete data. Indeed, the design of *recoverable* graph data augmentation [97, 112, 117] remains an open research area because is it difficult to determine *prima facie* what changes to a graph's topology or node features will preserve semantics. Moreover, as graphs are often abstract representations of structured data, it is also unclear what *invariances* are relevant to the downstream task. The assumption of a *separable* latent space may also be violated as intermediate points in this latent space may be meaningless in the discrete, structured input space. In contrast to natural image data, the systematic evaluation of these properties for graph SSL is difficult as it must accommodate both discrete and structured data.

Our Work. Better understanding the relationship between graph SSL practices and the aforementioned properties can help explain the behavior of existing frameworks and inform the design of new ones. Therefore, in this work, we take the first step by analyzing commonly used generic graph augmentations (GGAs) and designing useful tools that enable probing of these properties, including a theoretical framework and a synthetic data generation process that helps disentangle the effects of unrecoverable augmentations from performance. Our contributions can be summarized as follows:

Sec. 4.3: Analysis of Generalization and Separability. We provide the first generalization error bound for graph CL when using GGAs, demonstrating that GGAs can induce a performance-separability trade-off that is determined by underlying dataset properties (see Figure 4.1).

Sec. 4.4.1: Missing Invariance on Benchmark Datasets. On standard benchmarks, we show that models trained with GGAs have marginal improvements in accuracy and induce limited task-relevant invariance, at best, when compared to untrained encoders. We thus reveal a fundamental misalignment between the objectives and practical behavior of graph CL (see Figure 4.3).

Sec. 4.4.2: Synthetic Data Generation Process. We propose a synthetic data generation process that allows for control over augmentation recoverability and dataset separability (see Figure 4.2). Using this process, we validate our theoretical observations and demonstrate that recently proposed automated and implicit augmentation methods struggle to induce task-relevant invariances (see Figure 4.4).

4.2 Background & Related Work

In this section, we briefly discuss existing graph SSL paradigms. (Please see App. B.7 for additional discussion.) We then discuss the motivation behind data-centric properties (task-relevant invariance, separability and recoverabilty) central to this work.

Self-Supervised Graph Representation Learning. Recent advancements in representation learning have been driven by the SSL paradigm, where the goal is to ensure representations have high similarity between positive views of a sample and high dissimilarity between negative views. Existing SSL frameworks can be broadly categorized based on the mechanism adopted for enforcing this consistency: contrastive learning (CL) frameworks [54, 114, 124, 1, 97, 117, 53], such as GraphCL[1], use the InfoNCE loss; approaches that rely only on positive pairs, such as SimSiam [121] and BGRL [52] use Siamese architectures with stop gradient [121] and asymmetric branches [122] respectively; SpecCL [7] uses a spectral clustering loss (SpecLoss) to enforce consistency; others attempt to directly reduce redundancy between views [148, 151]. Despite these differences, all methods rely upon data augmentation to generate positive views, which are assumed to share semantics. Generic graph augmentations (GGAs) [1] are a popular strategy and assume limited changes to a graph's node features or topology are unlikely to alter its label. GGAs include random node dropping, edge perturbation, masking node attributes and sampling subgraphs. Other strategies include using diffusion matrices [51], GGAs with a non-uniform prior, automated methods which rely upon bi-level optimization [97] or adversarial optimization [117], and implicit methods, such as SimGRACE [53], which use weight-space perturbations as augmentations. Here, we primarily focus on GGAs due to their popularity, simplicity and effectiveness. Please see App. B.7 for additional discussion about augmentation paradigms.

Theoretical Analsyis of SSL. Several different perspectives have recently been used to successfully analyze SSL's behavior, including learning theory [7, 119, 152], causality [101, 100], information theory [120], and loss landscapes [153, 154, 155, 156]. Many of these analyses assume, either implicitly [101, 152] or explicitly [7, 150, 157, 158], the existence of a latent space that is *invariant* to augmentation functions and supports the properties of *recoverability* and *separability* (also see Figure 4.1).

Invariance to Augmentations: Producing similar representations for positive views, i.e., augmentations, induces invariance to the corresponding transformation function. Indeed, if augmentations are related by properties that are *not relevant* to the downstream task, representations will become *invariant* to this relationship over the course of SSL training and generalization will improve [159, 99]. Conversely, however, if augmentations induce invariance to *relevant* properties, then representations will fail to represent this information and are likely to lose task performance (e.g, color invariance is harmful when classifying different Labradors) [102, 79]. This latter point is often ignored by the theoretical analyses mentioned above. We note Tian et al.'s information theoretic framework [99] is a notable exception to this critique; we discuss the limitations of their results in App. B.3.

Recoverability and Separability: These properties state that in the latent space which instantiates the data generation process, two augmentations of a sample are close to each other (e.g., a clear and blurry dog) and unrelated points (e.g., dogs and cats) are sufficiently separated from each other. It is often implicitly assumed that only task-relevant augmentations are allowed [7, 150]. While originally proposed for manifolds [157], both recoverability and separability have been recently converted to graph connectivity properties [7] and verified empirically on modern deep learning methods [150]. Specifically, recoverability and separability can be used to bound generalization error on unseen data and we demonstrate how this can be done for graph CL in Sec. 4.3.

Notations. Let $\overline{\mathcal{X}}$ be a natural dataset with r different classes. Our use of word natural implies the samples in this dataset were collected via a natural sensing process (e.g., molecules from wet-lab experiments or scene graphs from images). We use $\mathcal{A}(.|\overline{\boldsymbol{g}})$ to denote the distribution of augmentations for the sample $\overline{\boldsymbol{g}} \in \overline{\mathcal{X}}$. Here, $\mathcal{A}(\boldsymbol{g}|\overline{\boldsymbol{g}})$ represents the probability of generating a particular augmentation $\boldsymbol{g}$, and $\mathcal{X} := \cup_{\overline{x} \in \mathcal{P}_{\overline{\mathcal{X}}}} \mathcal{A}(\cdot|\overline{\boldsymbol{g}})$ is the set of all samples transformed via our set of augmentation functions. Let $f : \mathcal{X} \to \mathbb{R}^d$ be a feature extractor, where $f(x)$ can be used for downstream tasks. Unless otherwise noted, let $\overline{\boldsymbol{g}}$ be a natural (graph) sample from $\overline{\mathcal{X}}$, $\mathcal{A}(\cdot|\cdot)$ be some GGA, and $\boldsymbol{g} \sim \mathcal{A}(\cdot|\overline{\boldsymbol{g}})$ be an augmented graph generated using a given GGA. $\mathcal{V}_{\overline{\boldsymbol{g}}}$ and $\mathcal{E}_{\overline{\boldsymbol{g}}}$ correspond, respectively, to the node and edge sets of $\overline{\boldsymbol{g}}$. We note our generalization analysis will specifically focus on the recently proposed contrastive loss by HaoChen et al. [7], called SpecLoss ($\mathcal{L}(f)$), which we define as follows: let $\boldsymbol{g} \sim \mathcal{A}(\cdot|\overline{\boldsymbol{g}})$, $\boldsymbol{g}^+ \sim \mathcal{A}(\cdot|\overline{\boldsymbol{g}})$, given $\overline{\boldsymbol{g}} \in \overline{\mathcal{X}}$, and $\boldsymbol{g}^- \sim \mathcal{A}(\cdot|\overline{\boldsymbol{g}}')$, given $\overline{\boldsymbol{g}}' \sim \mathcal{P}_{\overline{\mathcal{X}}} \wedge \overline{\boldsymbol{g}}' \neq \overline{\boldsymbol{g}}$. Then, for the positive/negative pairs $(\overline{\boldsymbol{g}}, \boldsymbol{g}^+)/(\overline{\boldsymbol{g}}, \boldsymbol{g}^-)$, SpecLoss is: $\mathcal{L}(f) = -2 \cdot \mathbb{E}_{\overline{\boldsymbol{g}}, \boldsymbol{g}^+}\left[f(\overline{\boldsymbol{g}})^\top f(\boldsymbol{g}^+)\right] + \mathbb{E}_{\overline{\boldsymbol{g}}, \boldsymbol{g}^-}\left[\left(f(\overline{\boldsymbol{g}})^\top f(\boldsymbol{g}^-)\right)^2\right]$. In a contemporary work, Saunshi et al. [119, 159] developed a generalization analysis for general contrastive loss functionals, including SpecLoss. Our analysis has a similar algorithmic flow as Saunshi et al.'s and hence the takeaways from our work can be easily extended for other contrastive methods as well. We provide additional discussion of this extension in App. B.1.

4.3 Generalization Bounds for CL with GGA

As discussed above, recent analyses have found that SSL generalization error can be bounded under the assumptions of invariance to relevant augmentations, recoverability, and separability.

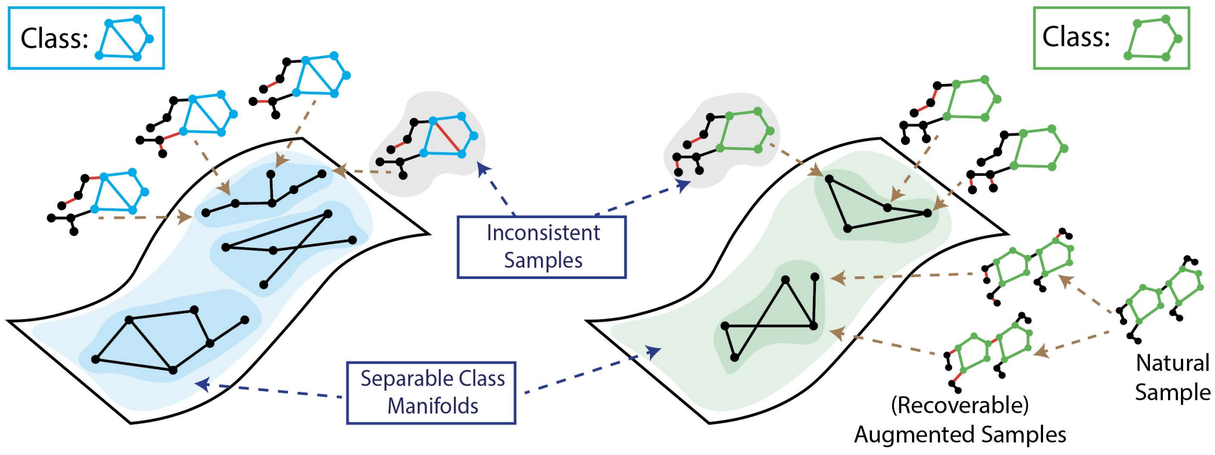


Figure 4.1: **Illustrating data-centric properties forming the core of our assumptions.** Our generalization analysis (Sec. 4.3) relies upon several data-centric properties, namely recoverability, separability, and frequency of inconsistent samples. Here, we illustrate these properties via a figure. (i) **Separability:** Samples from different classes should be *separable*, as illustrated by the existence of separate manifolds for different classes. This property helps assume the existence of a classifier h that can classify natural samples with low error. (ii) **Recoverability:** Labels of augmented samples should be *recoverable* from the original samples from which they were generated. This entails that augmentations generated from the same original samples are expected to be closer in latent space than two arbitrary samples, which will likely correspond to different classes. This property helps assume a constraint on the classifier h that it must also classify the augmentations of a sample to the same class as that of the sample. (iii) **Inconsistent Samples:** While the likelihood of generating augmentations that alter class semantics is low for image data, this if often note the case in graphs, especially when using generic graph augmentations. We refer to augmentations that can be generated from original samples belonging to different classes as *inconsistent*, and demonstrate that graph edit distance can be used to identify such samples. Overall, our theory shows inconsistent samples decrease separability and recoverability, harming generalization. (Figure inspired from Chung et al. [6] and HaoChen et al. [7].)

In this section, we demonstrate how GGAs influence these properties by deriving a generalization bound tailored for graph data. Notably, this bound allows us to demonstrate conditions where using GGAs results in low separability and recoverability, motivating the need for augmentations that induce task-relevant invariances that go beyond generic perturbative graph transformations.

Key Insight: Our main idea for the following analysis is that *GGAs can be instantiated in a general manner as a composition of graph edit operations.* This allows us to derive a unifying assumption related to recoverability and separability in terms of the graph edit distance (GED) between samples. Moreover, because GED amongst samples is a property intrinsic to the dataset, we can now discuss how the tightness of a SSL generalization error bound (SpecLoss's, specifically) will change as a function of GED between samples of underlying classes and augmentation strength.

We begin by defining GED and explaining how GGAs can be represented using graph edit operators.

Definition 1 (Graph Edit Distance)**.** *Let the elementary graph operators comprise the set of graph edits: these include node insertion, node deletion, edge deletion, edge addition, and an additional categorical feature replacement operator. Then,* $GED\left(g_1, g_2\right) = \min_{(e_1,\ldots,e_k)\in\mathcal{P}(g_1,g_2)} \sum_{i=1}^{k} c\left(e_i\right)$*, where* $\mathcal{P}\left(g_1, g_2\right)$ *is the set of paths (series of edit operations) that transforms graph* g_1 *to be isomorphic to graph* g_2*. Here,* e_i *is i-th edit operation in the path, and* $c(e_i)$ *is the cost for performing the edit.*

As shown in Table 4.1, frequently used GGA transforms can be easily decomposed using standard graph edit operators described in Def. 1. For example, assuming each operator has a unit cost, the edge perturbation augmentation can be seen as applying the minimum cost path consisting of edge deletion and edge addition operations to obtain $\boldsymbol{g}$ from $\overline{\boldsymbol{g}}$. Further, augmentation strength and the set of possible augmentations for a given natural sample can also be expressed in terms of GED:

Table 4.1: **Generic Graph Augmentations vs. Graph Edit Operators.** GGAs can be straightforwardly expressed using graph edit operators. Please see section B.4 for a detailed discussion.

Augmentations	**Graph Edit Operators**
Node Dropping	Node Deletion
Edge Perturbation	Edge Deletion, Edge Addition
Categorical Attribute Masking	Feature Masking Operator
Sub-graph Sampling	Node Deletions

Lemma 1. ***Allowable augmentations can be expressed using GED.*** *Let γ represent augmentation strength or the fraction of the graph that GGAs may modify. Then, $\delta \in \{\lfloor\gamma|\mathcal{V}_{\overline{\boldsymbol{g}}}|\rfloor, \lfloor\gamma|\mathcal{E}_{\overline{\boldsymbol{g}}}|\rfloor\}$ represents the number of discrete, allowable modifications for the specified GGA, so $GED(\overline{\boldsymbol{g}}, \boldsymbol{g}) \leq \delta$. Correspondingly, we have $\boldsymbol{g} \in \mathcal{A}(\overline{\boldsymbol{g}}) \Leftrightarrow GED(\boldsymbol{g}, \overline{\boldsymbol{g}}) \leq \delta$.*

For example, consider a graph $\boldsymbol{g} \sim \mathcal{A}(\cdot|\overline{\boldsymbol{g}})$, generated via node dropping. Then, $\boldsymbol{g}$ contains $1 - \delta$ nodes and the minimum cost path to transform $\overline{\boldsymbol{g}}$ to $\boldsymbol{g}$ contains only δ "node deletion" operations. Further, all augmentations generated from $\overline{\boldsymbol{g}}$ will have $1 - \delta$ nodes and thus have $GED(\overline{\boldsymbol{g}}, \boldsymbol{g}) \leq \delta$. In section B.4, we prove the above statement and demonstrate how to approximate $|\mathcal{A}(\overline{\boldsymbol{g}})|$ (e.g., the set of allowable augmentations for a given natural sample) using a combinatorial, counting procedure that is dependent on δ. Because GGAs are applied randomly, note that the probability of a generating a particular augmentation is $\mathcal{A}(\boldsymbol{g}|\overline{\boldsymbol{g}}) \approx \frac{1}{|\mathcal{A}(\overline{\boldsymbol{g}})|}$. Given these definitions, we now derive a unifying assumption in terms of GED between samples. We begin by formally introducing the separability and recoverability assumptions, focusing on the recently proposed, unified version [7]:

Assumption 1 (Separability plus Recoverability Assumption, (Assm. 3.5 in [7]))**.** *Let $\overline{\boldsymbol{g}} \in \overline{\mathcal{X}}$ and $y(\overline{\boldsymbol{g}})$ be its label, and $\boldsymbol{g} \sim \mathcal{A}(\cdot|\overline{\boldsymbol{g}})$. Assume that there exists a classifier h, such that $h(\boldsymbol{g}) = y(\overline{\boldsymbol{g}})$ with probability at least $1 - \alpha$. We refer to α as the error of h.*

See Figure 4.1 a visualization explaining this assumption. Intuitively, Assm. 1 states that there must exist a classifier h that is able to associate a sample's label with its augmentations, hence enabling recoverability, i.e., representations of augmentations are close to each other. Meanwhile, by ensuring augmentations of samples from a class with label "A" are classified as "A" and from a class with label "B" are classified as "B", the assumption simultaneously enables separability, i.e., representations of samples from different classes should be dissimilar. As we will see, the generalization bound will be a function of α, the probability that a classifier satisfying Assm. 1 associates augmentations of a class's samples with another class. As α grows larger, the generalization error bound becomes less tight. Therefore, it is important to understand how the choice of augmentation and augmentation strength (γ) can influence the error of h. We show one can also understand α as a trade-off between inter-class GED of samples and augmentation strength.

Intuitively, h will incur error on augmented samples that can be generated from a set of natural samples that belong to different underlying classes, as it is unclear how these samples should be embedded in a latent space. We now formally define such samples. First, using Lemma 1, we can determine if two augmentations could have been generated from the same sample.

Corollary 2. *(Co-occuring augmentations.) Let* $\overline{\boldsymbol{g}} \in \overline{\mathcal{X}}$ *and* $\boldsymbol{g}, \boldsymbol{g}' \in \mathcal{X}$*. Then,* $\boldsymbol{g} \sim \mathcal{A}(\overline{\boldsymbol{g}}) \wedge \boldsymbol{g}' \sim \mathcal{A}(\overline{\boldsymbol{g}}) \Leftrightarrow GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta$*, where* $\delta = \min\{\lfloor\gamma|\mathcal{V}_{\overline{\boldsymbol{g}}}|\rfloor, \lfloor\gamma|\mathcal{E}_{\overline{\boldsymbol{g}}}|\rfloor\ \lfloor\gamma|\mathcal{V}_{\boldsymbol{g}}|\rfloor, \lfloor\gamma|\mathcal{E}_{\boldsymbol{g}}|\rfloor\}$*.*

Given the above result, we now define inconsistent samples as follows.

Definition 3 (Inconsistent Samples)**.** *Let* $\boldsymbol{g} \in \mathcal{X}$*, and* $y : \overline{\mathcal{X}} \to r$ *be a labeling function. Further, let* $\overline{\mathcal{X}}_{in} = \{\overline{\boldsymbol{g}} | \overline{\boldsymbol{g}} \in \overline{\mathcal{X}} \wedge GED(\boldsymbol{g}, \overline{\boldsymbol{g}}) \leq \delta\}$ *be the set of natural samples that may have generated* $\boldsymbol{g}$ *and* $Y^*_{in} = \{y(\overline{\boldsymbol{g}}) | \overline{\boldsymbol{g}} \in \overline{\mathcal{X}}_{in}\}$ *be the set of unique labels. If* $\boldsymbol{g}$ *is an inconsistent sample,* $|Y^*_{in}| > 1$*.*

Essentially, if two augmentations co-occur (see Corr. 2) from two or more *different* natural samples, such that the samples *do not* share the same underlying label, we refer to such samples as *inconsistent* (also see Figure 4.1). Now, we assume the behavior of h on inconsistent samples is fixed such that $h(\boldsymbol{g}) = y$, for some fixed $y \in Y^*_{in}$. This allows us to use h to induce a r-way partition over $\mathcal{X}$, such that each sample, $\boldsymbol{g}$, belongs to a partition, $\mathbf{S}_h(\boldsymbol{g})$. Further, because h incurs error on inconsistent samples, α can be lower bounded by the ratio of inconsistent to total samples. To this end, we use GED to identify inconsistent samples by identifying disagreement amongst partitions as follows.

Lemma 2 (Using GED to identify inconsistent samples)**.** *Let* $\boldsymbol{g}, \boldsymbol{g}' \in \mathcal{X}$ *and* $GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta$ *such that* $\boldsymbol{g} \in \mathbf{S}_i \wedge \boldsymbol{g}' \in \mathbf{S}_j$ *and* $i \neq j$*, where partitions are induced by* h*. Then, at least one* $\tilde{\boldsymbol{g}} \in \{\boldsymbol{g}, \boldsymbol{g}'\}$ *must be an inconsistent sample.*

Note that the above lemma does not rely on ground-truth label information to identify inconsistent samples, but only GED from natural samples. Given that the error on inconsistent samples is irreducible, as it is unclear which $y \in Y_{in}$ is correct, we can lower bound the error of h as follows:

Corollary 4 (Error bound due to Inconsistent Samples)**.** *The error of* h *can be lower-bounded as*

$$\alpha \geq \frac{\sum_i^r \sum_{\boldsymbol{g} \in S_i, \boldsymbol{g}' \notin S_i} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta)}{|\mathcal{X}|}. \tag{4.1}$$

Here, the number of inconsistent samples can be approximated via $\sum_i^r \sum_{\boldsymbol{g} \in S_i, \boldsymbol{g}' \notin S_i} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta)$ and $|\mathcal{X}|$ can be estimated using a combinatorial counting procedure. Thus, the above corollary reflects the fact that error on inconsistent samples cannot be reduced due to label un-identifiability.

As mentioned before, the generalization bound by HaoChen et al. [7] for SpecLoss is a function of α. Deriving a lower bound on α will allow us to comment exactly when error is likely to become vacuous. To this end, we need a final definition of *partition dissimilarity* that induces a notion of clustering of similar datapoints in our analysis.

Definition 5 (Partition Dissimilarity)**.** *Let $S_1, \ldots, S_r$ be an r-way partition of $\mathcal{X}$. Then, we define the partition dissimilarity for a given partition as*

$$\phi_{\mathcal{X}}(S_i) = \frac{\sum_{\boldsymbol{g} \in S, \boldsymbol{g}' \notin S} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta)}{\sum_{\boldsymbol{g} \in S} |\{\boldsymbol{g}' | GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta\}|}. \tag{4.2}$$

Intuitively, we use the partitions induced by h as a proxy for class labels and co-occurrence as a notion of similarity (see Lemma 1). Then, the quality of the partition is determined by the ratio of the samples that belong to a given partition, but are also similar to samples from other partitions, to the total number of samples that are close to the partition. Note that partition dissimilarity is an often studied term in clustering problem and a general version of conductance, the property used for spectral clustering on a similarity graph which forms the basis of SpecLoss [7].

We are now ready to state our main result that re-derives the generalization error of SpecLoss in terms of GGAs, using the definitions of co-occurring pairs (Def. 2) and dissimilar partitions (Def. 5). Notably, we will decompose bound in terms of the number of co-occurring augmentation-pairs <u>within</u> the same partition and the number of pairs that cross partitions, which are defined respectively as, $\lambda = \sum_{\boldsymbol{g} \in S_*, \boldsymbol{g}' \in S_*} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta)$, and $\mu = \sum_{\boldsymbol{g} \in S_*, \boldsymbol{g}' \notin S_*} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta)$.

Theorem 6 (Generalization Bound for SpecLoss with GGA)**.** *Assume the representation dimension $k \geq 2r$ and Assm. 4 holds for $\alpha \geq 0$. Let F be a hypothesis class containing a minimizer f^*_{pop} of SpecLoss, $\mathcal{L}(f)$, which produces a $\lfloor k/2 \rfloor$-way partition of $\mathcal{X}$ denoted by $\{S_*\}$. Let its most dissimilar partition have dissimilarity denoted by $\rho_{\lfloor k/2 \rfloor} = \min_i \phi(S_i \in \{S_*\})$. Then, f^*_{pop} has a generalization error bounded as:*

$$\mathcal{E}(f^*_{pop}) \leq \widetilde{O}\left(\alpha / \rho^2_{\lfloor k/2 \rfloor}\right) = \widetilde{O}\left(\frac{r}{|\mathcal{X}|}\left[\mu + 2\lambda + \frac{\lambda^2}{\mu}\right]\right), \tag{4.3}$$

Discussion. By deriving expressions for α and ϕ as well as equivalently representing the original bound in terms of the more intuitive expressions, μ and λ, we can gain insights into several empirical and intuitive observations in graph CL. We will study these points further in Sec. 4.4.2 via a synthetic dataset that was motivated from the analysis above and allows more fine-grained evaluation.

Invariance and Relevance of Augmentations. GGAs assume that limited changes to a graph's structure will not alter its semantics and aggressively increasing augmentation strength will eventually harm generalization. However, through our analysis, we see that the generalization error bound is non-decreasing with respect to δ when $\frac{\lambda^2}{\mu} \leq \mu$, i.e., the number of <u>cross</u> partition pairs dominates the expression, as this ratio depends on δ. Indeed, for some $\delta' = \delta + \epsilon$, where $\epsilon > 0$, $\mu_{\delta'} = \sum_{\boldsymbol{g} \in S_i, \boldsymbol{g}' \notin S_i} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta) + \sum_{\boldsymbol{g} \in S_i, \boldsymbol{g}' \notin S_i} \mathbb{1}(2\delta \leq GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta + \epsilon) = \mu_\delta + \sum_{\boldsymbol{g} \in S_i, \boldsymbol{g}' \notin S_i} \mathbb{1}(2\delta \leq GED(\boldsymbol{g}, \boldsymbol{g}')) \leq 2\delta + \epsilon$. Thus, the number of cross partitions is always non-decreasing with respect to δ. Thus, *we clearly see that when augmentations are agnostic of the task, their corresponding invariances yield poor representations with vacuous generalization.*

Separability. Our analysis also demonstrates that the success of a particular augmentation strength is dependent on the GED between samples belonging to different classes. Given that inter-class GED is an intrinsic dataset property that proxies dataset separability, this implies that there are combinations of datasets and augmentation strengths for which GGAs will necessarily incur vacuous bounds, even for low augmentation strengths. In such settings, augmentations that improve recoverability and induce task-relevant invariances are necessary to improve downstream task performance. While many works have conjectured that task-relevant graph augmentations will improve performance, ours is the first to demonstrate why they are needed. Indeed, in Sec. 4.4.1, we find that GGAs are unable to induce such invariances on benchmark datasets.

Recoverability. As shown in Thm. 6, better recoverability will improve the tightness of the generalization bound. However, we see that from Coll. 4, that recoverability will only decrease as δ increases and as discussed above, there exist datasets where GGAs are not amenable. This further motivates the need for task-relevant augmentations so that the effects of poor augmentations are disentangled from method performance.

4.4 Experiments

In this section, we conduct experiments using both standard benchmarks (Sec. 4.4.1) and our proposed synthetic dataset generation process (Sec. 4.4.2) to empirically validate our theoretical conclusions.

4.4.1 A Closer Look at the Effectiveness of Invariance to GGA

In Sec. 4.3, we demonstrated GGAs can harm generalization by influencing recoverability and separability. Though computing these properties directly is intractable on benchmark datasets, our analysis for graph datasets and prior works on vision [100, 101, 102] show that if augmentations induce invariances that are *task-relevant*, downstream error should reduce. This corresponds to meaningfully related samples having similar representations (recoverable) and unrelated samples having dissimilar representations (separable). However, by using augmentations that perturb topology or features constrained to a small fraction of the original graph, existing graph SSL methods assume such perturbations are relevant to the downstream task. If this is the case, our analysis suggests we should see improvement in performance with increased invariance; else, we will witness no tangible correlation.

Experimental Setup: We evaluate seven graph SSL methods on seven, popular benchmark datasets. Specifically, we use the following representative algorithms: (i) *GraphCL* [1], a popular and effective graph CL method; (ii) *GAE*, Graph Autoencoder [160] that uses a reconstruction cost to learn representations; (iii) Augmentation-Augmented Autoencoder [161], which we adapt to graphs to create the Augmentation Augmented Graph Autoencoder (*AAGAE*) that minimizes the reconstruction error between the reconstruction for an augmented sample and the original; (iv) *SpecCL*, which uses the SpecLoss [7] for contrastive training; (v) *SimSiam* [121], a positive-sample-only framework that uses stop gradient; (vi) *BYOL* [122], which avoids negatives samples by using asymmetric branches alongside a stop gradient operation; and (vii) *Untrained representations*, which have been observed to be surprisingly competitive baselines for graph-based learning [160, 48, 117, 79]. To the best of our knowledge, ours is the first work to evaluate AAGAE and SpecCL for graph SSL. We use the same augmentations and encoder architecture as GraphCL. We add a straight-through estimator [162] to GAE/AAGAE's decoder for better training. See section B.6 for further details.

GGAs fail to induce task-relevant invariance on standard benchmarks. To measure whether augmentations have induced invariance, we measure recoverability using the representational similarity measures introduced by Wang and Isola [8]. Called Alignment and Uniformity, the two measures are a generalized version of the InfoNCE loss and also encompass other contrastive losses, such as SpecLoss. Formally, alignment is defined as: $\mathcal{L}_{\text{align}}(f; \mathcal{A}) \triangleq \mathbb{E}_{(\boldsymbol{g},\boldsymbol{g}') \sim \mathcal{A}(\cdot|\overline{\boldsymbol{g}})}\left[\|f(\boldsymbol{g}) - f(\boldsymbol{g}')\|_2^2\right]$. To determine if the invariance is task-relevant, we determine if improved alignment is indicative of improved performance with respect to an untrained baseline model.

Results. Fig. 4.3 shows the difference in invariance and kNN with respect to an untrained model's accuracy, averaged over 10 seeds. As can be seen, there is not noticeable correlation between invariance and accuracy, especially with respect to the untrained baseline. Notably, on the Reddit dataset, all methods have improved invariance, but do not have significantly better kNN accuracy. Overall, this experiment demonstrates that learning invariance to GGAs is both difficult and often unrelated to task performance, clearly indicating the GGAs struggle to induce task-relevant invariances and do not support recoverable, separable latent spaces needed for good generalization. Moreover, given that GGAs have unknown recoverability on

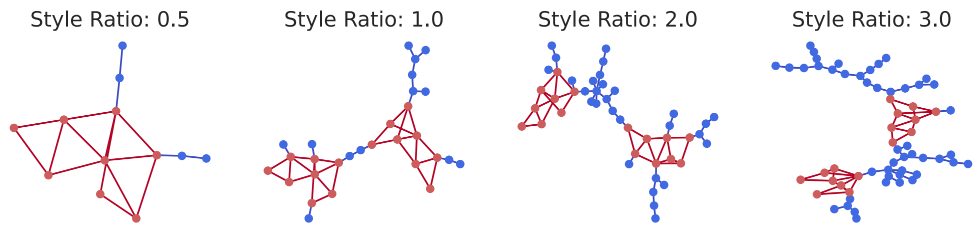


Figure 4.2: **Synthetic Dataset Generation.** A class-specific motif completely determines the label, and is therefore considered "content". To vary the amount of style, the size of the background tree graph is a ratio of the number of "content" nodes. Our dataset goes beyond binary benchmarks and allows for content-aware augmentations, a critical component to understanding graph SSL.

standard datasets, and that trained models were not able to sufficiently outperform untrained baselines, there is need for new datasets where it is possible to go beyond GGA and where we can better understand the merits of different graph SSL paradigms.

4.4.2 Evaluating Graph SSL Methods in a Controlled Setting

Our analysis indicates the role played by recoverability and separability under task-relevant invariances dramatically influences generalization performance. However, given our results that GGAs do not enable these properties and the fact that task-relevance is difficult to define on existing benchmark datasets, empirical verification of our claims requires a dataset that directly enables control over the data generation process. We thus introduce a synthetic dataset that allows us to illustrate how invariance and class separability must be jointly considered when designing augmentations.

4.4.2.1 Synthetic Data Generation Process

Given that standard benchmark datasets and augmentation practices are uninformative when evaluating the recoverability and invariance of augmentations, we propose a synthetic data generation process that allows us to understand how the data-dependent assumptions of SSL hold for graph datasets. This process not only enables oracle augmentations where recoverability is known, but also allows us some control over dataset separability. Our synthetic dataset generation process is designed in accordance to a latent variable model which assumes that the underlying data generation latent representation space can be partitioned into *style* and *content*. Here, *style* represents information that is irrelevant to the downstream task and can be perturbed (i.e., augmented) without changing sample semantics, while *content* represents task-relevant information and should be preserved. We note that while von Kügelgen et al. [100] used the same latent variable model to demonstrate that SSL with data augmentation is able to recover features which disentangle *style* vs. *content*, our objective for using this perspective is to develop a grounded benchmark that provides adjustable knobs over content (task-relevant) and style (task-irrelevant) information. These knobs allow us to understand how data-centric properties affect the performance of different graph SSL algorithms (see Fig. 4.4). While designing content-aware augmentations for arbitrary graph datasets is a hard problem [79], with oracle knowledge of the data generation process, we can evaluate content-aware augmentations (CAAs) with high recoverability at varying levels of separability, which we approximate through different style levels.

Generation Process: The proposed data generation process has three components: a set of C motifs, $\mathcal{M}$, that uniquely determine C classes; a random graph generator, $RBG(n)$, parameterized by the number of nodes (we can equivalently define this based on number of edges); and κ, the style multiplier, which controls how much irrelevant information a sample contains. To generate a sample, we attach a randomly generated background graph (*i.e.*, style component) to a motif (*i.e.*, content) according to the style multiplier. This simple process addresses several limitations often encountered in graph CL evaluation. Specifically, it (i) allows for varying levels of content-aware augmentation (*i.e.*, edges that can be perturbed in the background graph without harming the motif); (ii) is easily extended beyond binary classification; (iii) contains relatively large number of samples; and (iv) offers a natural test bed for GNN size generalization or transfer learning [18].

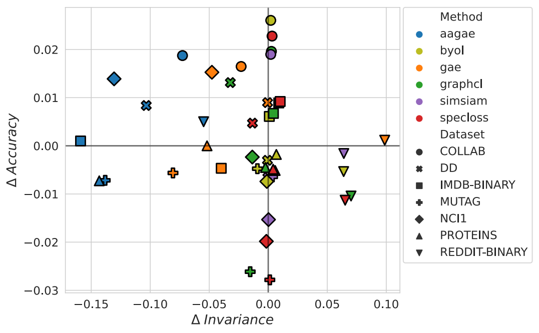


Figure 4.3: **Invariance vs. KNN Acc.** The change in invariance (Inv.) and accuracy w.r.t. to an untrained model is plotted, where Inv. is measured according to [8]. We see: Inv. has not significantly increased for many datasets/methods, improved Inv. does not necessarily entail better performance (see Reddit), and AAGAE/GAE often sees decreased Inv., likely due to use of a decoder.

4.4.2.2 Difficulties in Recovering Style Invariant Representations

Several real graph datasets can be understood through a style vs. content perspective. For example, in drug discovery tasks [20], molecules can be split into functional groups (content) and carbon rings or scaffold structure (style). One may thus ask: how does varying levels of style vs. content affect the performance of graph URL algorithms, and how do different algorithms benefit from the use of content-aware augmentations? To answer these questions,

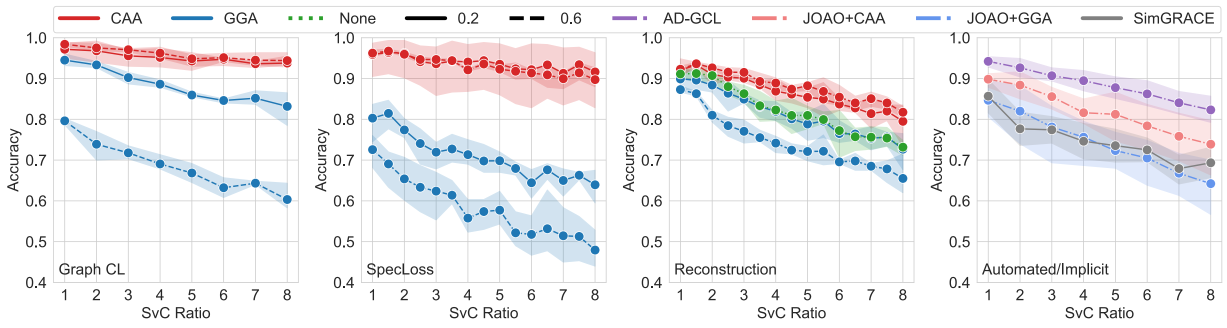


Figure 4.4: **Style Invariance over Paradigms.** We evaluate several SSL algorithms with different augmentation paradigms and changing style vs. content ratios. We find several notable results: (i) CAAs induce style invariance in contrastive methods, but GGAs do not; (ii) reconstruction methods do not recover task-relevant invariances, even when using CAAs; and (iii) advanced augmentations methods (AD-GCL, JOAO, SimGRACE) lose performance as style increases, indicating they do not induce style-invariance.

we conduct the following experiment:

Experimental Setup. Let $C = 6$, $\kappa = 4$ and define $RBG(n)$ through a random tree generator, where n is number of the nodes belonging the motif, scaled by κ. Node features are a constant 10-dimensional vector. To increase task difficulty, we randomly insert between 1-3 motif copies into each sample. Using the specified instaniation of the generation process, we train GraphCL, AAGAE, GAE, and SpecLoss with *content-preserving* edge dropping and random edge dropping at 20% and 60% augmentation strength. We also evaluate two recently proposed automated augmentation methods, JOAO [97] and AD-GCL[117], as well as SimGRACE [53], which uses implicit, weight space perturbations. JOAO is trained with a GGA prior and an expanded GGA prior that includes content-preserving edge dropping. AD-GCL is trained using a learnable edge-dropping augmentor. A 5-layer GIN encoder is used and models are trained for 60 epochs using Adam (with a learning rate of 0.01). After training, all models are evaluated using the linear probe protocol [54] at varying style ratios. Given that style information is not relevant to the downstream task, we expect models that have truly learned invariance to this information will retain strong performance across different ratios. See section B.6 for more model and training details.

Results. We make the following observations using Fig. 4.4, which clearly demonstrate the value of the proposed benchmark in studying the behavior of different SSL and augmentation paradigms. (i) In accordance to Sec. 4.3, we empirically see that both GraphCL and SpecLoss do not loss performance as the style ratio increases when using CAAs, indicating the model has learned task-relevant invariances. (ii) Auto-encoding reconstruction methods are an alternative SSL paradigm, but unfortunately also struggle to recover style-invariant solutions. Moreover, the use of the CAAs with such methods does not improve performance as effectively as in contrastive paradigms. (iii) For the first time, we are able to evaluate whether automated methods, which aim to recover strong augmentations without expensive hyper-parameter tuning or hand designing, are able to recover an optimal augmentation that generalizes across style ratios.

Unfortunately, we see both AD-GCL [117] and JOAO [97] lose performance as the style ratio increases, indicating such a solution has not been found. Indeed, JOAO is unable to find such a solution even when the augmentation prior includes the oracle CAAs. These results not only highlight the brittleness of such automated methods, but indicate our benchmark is a necessary testbed for such methods. (iv) To avoid corrupting graph semantics when using input-space augmentations, SimGRACE [53] instead uses implicit, weight-space augmentations. However, we find, despite tuning the perturbation parameter, SimGRACE cannot recover strong, style-invariant performance. Overall, using our grounded synthetic benchmark, we are not only able to able to compare the performance of graph SSL algorithms when data-centric properties are supported (e.g., recoverable augmentations), but are also able to identify limitations of advanced augmentation methods that were not apparent using standard benchmarks.

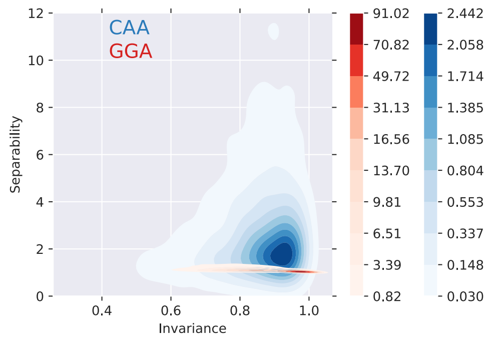


Figure 4.5: **Invariance vs. Separability**. On our synthetic data with style-to-content ratio $\kappa = 6$ and 20% augmentation strength, GraphCL trained with random augmentations produces representations with high invariance but low separability. In contrast, using content preserving augmentations leads to almost as high invariance, but much greater separability.

4.4.2.3 Invariance vs. Separability

We now use our synthetic benchmark to investigate how augmentation recoverability influences the balance of invariance and separability in the learned latent space. Considered in isolation, invariance can be trivially satisfied through representation collapse, i.e., all samples are mapped to highly similar representations. However, such representations are not separable as they cannot meaningfully distinguish classes. Therefore, in the following experiment, we jointly consider these properties to understand the benefits of CAAs.

Experimental Setup. Using a synthetic dataset at $\kappa = 6$, we respectively train GraphCL with *content-preserving* and random edge dropping at 20% augmentation strength. We compute an invariance score for each natural sample by computing the average cosine similarity of its representation with that 30 different augmentations. We compute a separability score by dividing the maximum cosine similarity to a sample of the same class by the maximum cosine similarity to a sample of another class.

Results. Figure 4.5 shows kernel density estimates of the number of samples that have a given invariance and separability, when training with GGA or CAA. GGA induces representations with somewhat higher invariance but much lower separability scores, suggesting some representation collapse are occurred. Indeed, with a higher augmentation strength (60%), we found that using GGA produced invariance and separability scores very close to 1 for all samples, indicating strong collapse. On the other hand, CAA helps GraphCL achieve over an order of magnitude higher separability and still preserves comparably high invariance. We observed similar trends for SpecLoss.

Invariance vs. Separability in Realistic Settings. In App. B.2, we replicate this experiment using BACE [10], a molecule-protein interaction dataset, and the biochemistry-based augmentations proposed by Sun et al. [11] as CAAs. We find that our observations continue to hold in this real-world use-case, demonstrating the generality of our theory and practicality of our synthetic benchnmark.

4.5 Conclusion

In this work, we rigorously contextualize, theoretically and empirically, the role of data-dependent properties for graph CL. We propose a novel generalization analysis which, for the first time, formalizes the limitations of using GGAs in graph CL. As we note in Sec. 4.3, our results can be extended to other contrastive frameworks by leveraging our insight on representing graph augmentations as composable graph-edit operations and extending the contemporary work of Saunshi et al. [159]. We suspect a similar extension can also be made for predictive methods like BYOL by using the analysis of Wei et al. [150] (see App. B.1 for further discussion). In line with our theory, we empirically demonstrate that GGAs fail to induce useful task-relevant invariances on standard benchmarks. We note our empirical results already demonstrate the generality of our results across different methods. Moreover, our insights motivate the design of a principled synthetic benchmark that provides a controlled setting for studying the role of data-dependent properties in graph SSL. Our benchmark also serves as a useful testbed for evaluating the abilities of automated or implicit augmentations techniques. Given the shortcomings we illustrate for such methods on synthetic datasets, we argue the development of domain specific strategies [11] may be a more fruitful direction for future work.

Part II: Uncertainty Estimation with Graph Neural Networks

CHAPTER 5

Accurate Estimation of Epistemic Uncertainty for GNNs

5.1 Introduction

Having discussed how to improve representation expressivity in Part I, we turn to the equally important problem of ensuring that these representations and models are trust-worthy and safe. To this end, in Part II, we focus on improving the uncertainty estimation of the GNN-based models, as this property underlies several key safety tasks. Indeed, as GNNs are increasingly deployed in critical applications with test-time distribution shifts [22, 163, 164, 92, 165], it becomes necessary to expand model evaluation to include safety-centric metrics, such as calibration errors [34], out-of-distribution (OOD) rejection rates [62], and generalization error predictions (GEP) [61], to holistically understand model performance in such shifted regimes [166, 167]. Notably, improving on these additional metrics often requires reliable uncertainty estimates, such as maximum softmax or predictive entropy, which can be derived from prediction probabilities. Although there is a clear understanding in the computer vision literature that the quality of uncertainty estimates can noticeably deteriorate under distribution shifts [168, 169], the impact of such shifts on graph neural networks (GNNs) remains relatively under-explored.

Post-hoc calibration methods [34, 170, 171, 172], which use validation datasets to rescale logits to obtain better calibrated models, are an effective, accuracy-preserving strategy for improving uncertainty estimates and model trust-worthiness. Indeed, several post-hoc

The material in this chapter is derived from the paper "Accurate and Scalable Estimation of Epistemic Uncertainty for Graph Neural Networks" [81], which appeared in the proceedings of the International Conference on Learning Representations 2024. Code can be accessed here.

calibration strategies [42, 41] have been recently proposed to explicitly account for the non-IID nature of node-classification datasets. However, while these methods are effective at improving uncertainty estimate reliability on in-distribution (ID) data, they have not been evaluated on OOD data, where they may become unreliable. To this end, training strategies which produce models with better intrinsic uncertainty estimates are valuable as they will provide better out-of-the-box ID and OOD estimates, which can then be further combined with post-hoc calibration strategies if desired.

The ΔUQ training framework [173] was recently proposed as a scalable, single model alternative for vision models ensembles and has achieved state-of-the-art performance on calibration and OOD detection tasks. Central to ΔUQ's success is the concept of *anchored* training, where models are trained on stochastic, relative representations of input samples in order to simulate sampling from different functional modes at test time (Sec. 5.2.) While, on the surface, ΔUQ also appears as a potentially attractive framework for obtaining reliable, intrinsic uncertainty estimates on graph-based tasks, there are several challenges that arise from the structured, discrete, and variable-sized nature of graph data that must be resolved first. Namely, the anchoring procedure used by ΔUQ is not applicable for graph datasets, and it is unclear how to design alternative anchoring strategies such that sufficiently diverse functional modes are sampled at inference to provide reliable epistemic uncertainty estimates.

Proposed Work. Thus, our work proposes G-ΔUQ, a novel training paradigm which provides better intrinsic uncertainty estimates for both graph and node classification tasks through the use of newly introduced graph-specific, anchoring strategies. Our contributions can be summarized as follows:

- **(Partially) Stochastic Anchoring for GNNs.** We propose G-ΔUQ, a novel training paradigm that improves the reliability of uncertainty estimates on GNN-based tasks. Our novel graph-anchoring strategies support partial stochasticity GNNs as well as training with pretrained models. (Sec. 5.3).
- **Evaluating Uncertainty-Modulated CIs under Distribution Shifts.** Across covariate, concept and graph-size shifts, we demonstrate that G-ΔUQ leads to better calibration. Moreover, G-ΔUQ's performance is further improved when combined with post-hoc calibration strategies on several node and graph-level tasks, including new safety-critical tasks (Sec. 5.5).
- **Fine-Grained Analysis of G-ΔUQ.** We study the calibration of architectures of varying expressivity and G-ΔUQ 's ability to improve them under varying distribution shift. We further demonstrate its utility as a lightweight strategy for improving the calibration of pretrained GNNs (Sec. 5.6).

5.2 Background & Related Work

While uncertainty estimates are useful for a variety of safety-critical tasks [62, 61, 34], DNNs are well-known to provide poor uncertainty estimates directly out of the box [34]. To this end, there has been considerable interest in building calibrated models, where the confidence of a prediction matches the probability of the prediction being correct. Notably, since GEP and OOD detection methods often rely upon transformations of a model's logits, improving calibration can in turn improve performance on these tasks as well. Due to their accuracy-preserving properties, post-hoc calibration strategies, which rescale confidences after training using a validation dataset, are particularly popular. Indeed, several methods [34, 170, 171, 172] have been proposed for DNNs in general and, more recently, dedicated node-classifier calibration methods [42, 41] have also been proposed to accommodate the non-IID nature of graph data. (See App. C.9 for more details.) Notably, however, such post-hoc methods do not lead to reliable estimates under distribution shifts, as enforcing calibration on ID validation data does not directly lead to reliable estimates on OOD data [169, 168, 174].

Alternatively, Bayesian methods have been proposed for DNNs [175, 176], and more recently GNNs [177, 178], as inherently "uncertainty-aware" strategies. However, not only do such methods often lead to performance loss, require complicated architectures and additional training time, they often struggle to outperform the simple Deep Ensembles (DEns) baseline [179]. By training a collection of independent models, DEns is able to sample different functional modes of the hypothesis space, and thus, capture epistemic variability to perform uncertainty quantification [180]. Given that DEns requires training and storing multiple models, the SoTA ΔUQ framework [173] was recently proposed to sample different functional modes using only a single model, based on the principle of *anchoring*.

Background on Anchoring. Conceptually, anchoring is the process of creating a relative representation for an input sample in terms of a random "anchor." By randomizing anchors throughout training (e.g., stochastically centering samples with respect to different anchors), Δ-UQ emulates the process of sampling and learning different solutions from the hypothesis space.

In detail, let $\mathcal{D}_{train}$ be the training distribution, $\mathcal{D}_{test}$ be the testing distribution, and $\mathcal{D}_{anchor} := \mathcal{D}_{train}$ be the anchoring distribution. Existing research on stochastic centering has focused on vision models (CNNs, ResNets, ViT) and used input space transformations to construct anchored representations. Specifically, given an image sample with corresponding label, $(\mathbf{I}, y)$, and anchor $\mathbf{C} \in \mathcal{D}_{anchor}$, anchored samples were created by subtracting and

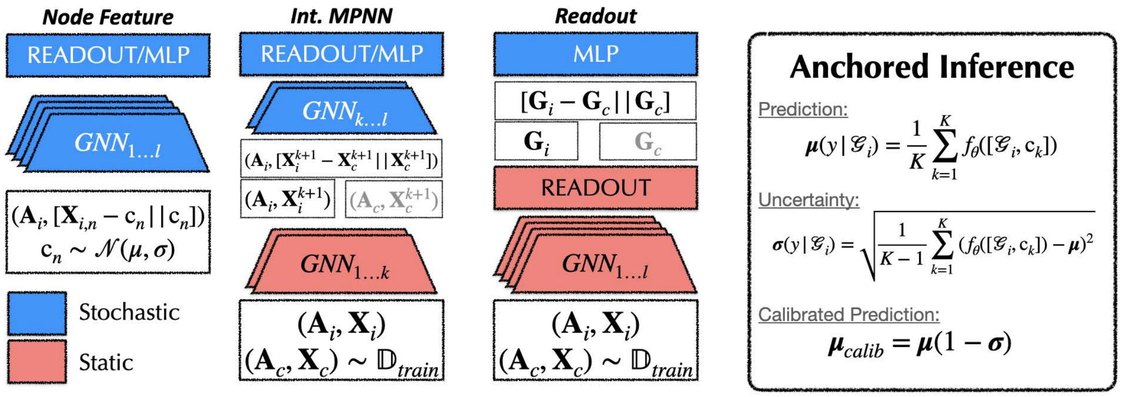

Node Feature
READOUT/MLP
$GNN_{1...l}$
$(\mathbf{A}_i, [\mathbf{X}_{i,n} - c_n || c_n])$
$c_n \sim \mathcal{N}(\mu, \sigma)$
Stochastic
Static
Int. MPNN
READOUT/MLP
$GNN_{k...l}$
$(\mathbf{A}_i, [\mathbf{X}_i^{k+1} - \mathbf{X}_c^{k+1} || \mathbf{X}_c^{k+1}])$
$(\mathbf{A}_i, \mathbf{X}_i^{k+1})$
$(\mathbf{A}_c, \mathbf{X}_c^{k+1})$
$GNN_{1...k}$
$(\mathbf{A}_i, \mathbf{X}_i)$
$(\mathbf{A}_c, \mathbf{X}_c) \sim \mathbb{D}_{train}$
Readout
MLP
$[\mathbf{G}_i - \mathbf{G}_c || \mathbf{G}_c]$
$\mathbf{G}_i$
$\mathbf{G}_c$
READOUT
$GNN_{1...l}$
$(\mathbf{A}_i, \mathbf{X}_i)$
$(\mathbf{A}_c, \mathbf{X}_c) \sim \mathbb{D}_{train}$
Anchored Inference
Prediction:
$\boldsymbol{\mu}(y | \mathcal{G}_i) = \frac{1}{K}\sum_{k=1}^{K} f_\theta([\mathcal{G}_i, c_k])$
Uncertainty:
$\boldsymbol{\sigma}(y | \mathcal{G}_i) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K} (f_\theta([\mathcal{G}_i, c_k]) - \boldsymbol{\mu})^2}$
Calibrated Prediction:
$\boldsymbol{\mu}_{calib} = \boldsymbol{\mu}(1 - \boldsymbol{\sigma})$

Figure 5.1: **Overview of G-ΔUQ models.** Here, we present a conceptual overview of how G-ΔUQ induces partially stochastic models. This figure is complementary to C.1.

then channel-wise concatenating two images: $[\mathbf{I} - \mathbf{C} || \mathbf{C}]^*$. Given the anchored representation, a corresponding stochastically centered model can be defined as $f_\theta : [\mathbf{I} - \mathbf{C} || \mathbf{C}] \rightarrow \hat{y}$, and can be trained as shown in Fig. 5.2. At inference, similar to ensembles, predictions and uncertainties are aggregated over different hypotheses. Namely, given K random anchors, the mean target class prediction, $\boldsymbol{\mu}(y|\mathbf{I})$, and the corresponding variance, $\boldsymbol{\sigma}(y|\mathbf{I})$ are computed as: $\boldsymbol{\mu}(y|\mathbf{I}) = \frac{1}{K}\sum_{k=1}^{K} f_\theta([\mathbf{I} - \mathbf{C}_k, \mathbf{C}_k])$ and $\boldsymbol{\sigma}(y|\mathbf{I}) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}(f_\theta([\mathbf{I} - \mathbf{C}_k, \mathbf{C}_k]) - \boldsymbol{\mu})^2}$. Since the variance over K anchors captures epistemic uncertainty by sampling different hypotheses, these estimates can be used to modulate the predictions: $\boldsymbol{\mu}_{\text{calib.}} = \boldsymbol{\mu}(1 - \boldsymbol{\sigma})$. Notably, the rescaled logits and uncertainty estimates have led to state-of-the-art performance on image outlier rejection, calibration, and extrapolation [181, 182].

5.3 Graph-ΔUQ: Uncertainty-Aware Predictions

Given Δ-UQ's success in improving calibration and generalization [182] under distribution shifts on computer vision tasks and the limitations of existing post-hoc strategies, stochastic centering appears as a potentially attractive framework for obtaining reliable uncertainty estimates when performing GNN-based classification tasks. However, there are several challenges that must be addressed before to applying it to graph data. Namely, while input space transformations, which induce fully stochastic models, were sufficient for sampling diverse functional hypotheses from vision models, it is (i) non-trivial to define such transformations when working with variable sized, discrete graph data and (ii) unclear whether full stochasticity is in fact needed when working with message passing models. Below, we explore these issues through novel graph anchoring strategies. However, we begin with a conceptual

*For example, channel wise concatenating two RGB images creates a 6 channel sample.

discussion of the role of anchoring strategies in generating reliable uncertainty estimates.

What are the goals of anchoring?: As discussed in Sec. 5.2, epistemic uncertainty can be estimated by aggregating the variability over different functional hypotheses [183]. Indeed, the prevalent wisdom behind the success of DeepEns is its ability to sample *diverse* functional hypotheses. Since these hypotheses are more likely to differ on OOD inputs, aggregating them can lead to better generalization and uncertainity estimates. Insofar as stochastic centering seeks to simulate an ensemble through a single model, a key goal of the anchoring distribution/strategy is then to ensure that sampled hypotheses are also diverse. [173] obtained sufficient diversity by using input space anchoring to sample a fully stochastic network. However, in the context of Bayesian neural networks (BNNs), it was recently shown that partial stochasticity can perform equally well with respect to fully stochastic BNNs at significantly less cost [184]. This suggests that in addition to the "amount" of diversity, the "effective" or functional diversity is also important for performance. However, in practice, it is difficult to control this balance, so existing methods default to heuristics that only promote diverse hypotheses. For example, DeepEns uses different random seeds or shuffles the batch order when creating ensemble members, and Δ-UQ relies upon fully stochastic models. To this end, we propose three different anchoring strategies that only handle the difficulties of working with graph data and GNNs, but also induce different scales of the aforementioned balance. At a high-level, our strategies trade-off the amount of stochasticity (i.e., amount of diversity) and the semantic expressivity of the anchoring distribution to accomplish this.

```
#training
for (I,y) in trainloader:
  C = create_anchors(n=BatchSize)
  I_Anc = CONCAT([I-C,C],dim=1)
  preds = model(I_Anc)
  loss = criterion(preds,Y)
#inference
anc_preds = []
testAnc = create_anchors(n=K)
for A in testAncs:
  I_anc = CONCAT([I-A,A],dim=1)
  preds = model(I_anc)
  anc_preds.append(preds)
P = CONCAT(anc_preds,dim=0)
mu = MEAN(P,dim=0)
var = STDDEV(P,dim=0)
```

Figure 5.2: **Training and Inference with Anchoring.**

Notations. Let $\mathcal{G} = (\mathbf{X}^0, \mathbf{A}, Y)$ be a graph with node features $\mathbf{X}^0 \in \mathbb{R}^{N\times d}$, adjacency matrix $\mathbf{A} \in [0,1]^{N\times N}$ and labels Y, where N, d, q denote the number of nodes, feature dimension and number of classes, respectively. When performing graph classification, $Y \in \{0,1\}^q$; for node classification, let $Y \in \{0,1\}^{N\times q}$.

We define a graph classification GNN consisting of ℓ message passing layers (MPNN), a graph-level readout function ($\texttt{READOUT}$), and classifier head ($\texttt{MLP}$) as follows: $\mathbf{X}^{\ell+1} = MPNN^{\ell+1}\left(\mathbf{X}^{\ell}, \mathbf{A}\right)$, $\mathbf{G} = \texttt{READOUT}\left(\mathbf{X}^{\ell+1}\right)$, and $\hat{Y} = \texttt{MLP}\left(\mathbf{G}\right)$ where $\mathbf{X}^{\ell+1} \in \mathbb{R}^{N\times d_\ell}$ is the intermediate node representation at layer $\ell+1$, $\mathbf{G} \in \mathbb{R}^{1\times d_{\ell+1}}$ is the graph representation, and $\hat{Y} \in \{0,1\}^q$ is the predicted label. When performing node classification, we do not include the $\texttt{READOUT}$ layer, and instead output node-level predictions: $\hat{Y} = \texttt{MLP}\left(\mathbf{X}^{\ell+1}\right)$. We use subscript $_i$ to indicate indexing and $||$ to indicate concatenation.

We begin by introducing a graph anchoring strategy for inducing fully stochastic GNNs. Due to size variability and discreteness, performing a structural residual operation by subtracting two adjacency matrices would be ineffective at inducing an anchored GNN. Indeed, such a transform would introduce artificial edge weights and connectivity artifacts. Likewise, when performing graph classification, we cannot directly anchor over node features, since graphs are different sizes. Taking arbitrary subsets of node features is also inadvisable as node features cannot be considered IID. Further, due to iterative message passing, the network may not be able to converge after aggregating l hops of stochastic node representations (see C.15 for details). Furthermore, there is a risk of exploding stochasticity when anchoring MPNNs. Namely, after l rounds of message passing, a node's representations will have aggregated information from its l hop neighborhood. However, since anchors are unique to individual nodes, these representations are not only stochastic due to their own anchors but also those of their neighbors.

To address both these challenges, we instead fit a d-dimensional Gaussian distribution over the training dataset's input node features which is then used as the anchoring distribution (see Fig. 5.3). While a simple solution, the fitted distribution allows us to easily sample anchors for arbitrarily sized graphs, and helps manage stochasticity by reducing the complexity of the anchoring distribution, ensuring that overall stochasticity is manageable, even after aggregating the l-hop neighborhood. (See C.15 for details.)

```
#Node Classification
#(N,inputdim)
X,adj = get_node_dataset()

#Graph Classification
GraphData = get_graph_dataset()
#(Total Num Nodes,inputdim)
X = CONCAT([g.x for g in GraphData],dim=0)

#Create Anchoring Dist
mu = X.mean(dim=0) #(1,inputdim)
std = X.std(dim=0) #(1,inputdim)
#inputdim dimensional Gaussian
AncDist = Normal(mu=mu, std=std)

#Create TRAINING Anchors
#(N,inputdim)
C_Node = AncDist.sample(N)
#(num_nodes(g),inputdim)
C_Graph = AncDist.sample(g.X.shape[0])

#Create INFERENCE Anchors
#(N,inputdim)
C_Node = AncDist.sample(1).repeat(N,dim=0)
#(num_nodes(g),inputdim)
C_Graph =
AncDist.sample(1).repeat(g.X.shape[0],dim=0)

#Anchored Representations
AncNode = CONCAT([X-C_Node, C_Node],dim=1)
AncGraph = CONCAT([X-C_Graph, C_Graph],dim=1)
```

Figure 5.3: **Node Feature Anchoring Pseudocode.**

We emphasize that this distribution is only used for anchoring and does not assume that the dataset's node features are normally distributed. During training, we randomly sample a unique anchor for each node. Mathematically, given anchors $\mathbf{C}^{N\times d} \sim \mathcal{N}(\mu, \sigma)$, we create the anchored node features as: $[\mathbf{X}^0 - \mathbf{C}||\mathbf{X}^0]$. During inference, we sample a fixed set of K anchors and compute residuals for all nodes with respect to the *same* anchor after performing appropriate broadcasting, e.g., $\mathbf{c}^{1\times d} \sim \mathcal{N}(\mu, \sigma)$, where $\mathbf{C} := \texttt{REPEAT}(\mathbf{c}, N)$ and $[\mathbf{X}^0 - \mathbf{C}_k||\mathbf{X}^0]$ is the kth anchored sample. For datasets with categorical node features, anchoring can be performed after embedding the node features into a continuous space. If node features are not available, anchoring can still be performed via positional encodings [185], which are known to improve the expressivity and performance of GNNs [186]. Lastly, note that performing node feature anchoring (NFA) is the most analogous extension of Δ-UQ to graphs as it results in fully stochastic GNNs. This is particularly true on node classification tasks where each node can be viewed as an individual sample, similar to a image sample original ΔUQ formulation.

5.3.1 Hidden Layer Anchoring for Graph Classification

While NFA can conceptually be used for graph classification tasks, there are several nuances that may limit its effectiveness. Notably, since each sample (and label) is at a graph-level, NFA not only effectively induces multiple anchors per sample, it also ignores structural information that may be useful in sampling more *functionally diverse* hypotheses, e.g., hypotheses which capture functional modes that rely upon different high-level semantic, non-linear features. To improve the quality of hypothesis sampling, we introduce hidden layer anchoring below, which incorporates structural information into anchors at the expense of full stochasticity in the network (See Fig. 5.2):

Hidden Layer and Readout Anchoring: Given a GNN containing ℓ MPNNlayers, let $2 \leq r \leq \ell$ be the layer at which we perform anchoring. Then, given the intermediate node representations $\mathbf{X}^{r-1} = MPNN^{r-1}(\mathbf{X}^{r-2}, \mathbf{A})$, we randomly shuffle the node features over the entire batch, ($\mathbf{C} = \text{SHUFFLE}(\mathbf{X}^{r-1}, \text{dim} = 0)$), concatenate the residuals ($[\mathbf{X}^{r-1} - C||C]$), and proceed with the `READOUT` and `MLP` layers as usual. (See C.2 for corresponding pseudocode.) Note the gradients of the query sample are not considered when updating parameters, and $MPNN^r$ is modified to accept inputs of dimension $d_r \times 2$ (to take in anchored representations as inputs). At inference, we subtract a single anchor from all node representations using broadcasting. Hidden layer anchoring induces the following GNN: $\mathbf{X}^{r-1} = MPNN^{r-1}(\mathbf{X}^{r-2}, \mathbf{A})$, $\mathbf{X}^{r} = MPNN^{r}\left([\mathbf{X}^{r-1} - \mathbf{C}||\mathbf{C}], \mathbf{A}\right)$, and $\mathbf{X}^{\ell+1} = MPNN^{r+1\ldots\ell}\left(\mathbf{X}^{r}, \mathbf{A}\right)$, and $\hat{Y} = \texttt{MLP}(\texttt{READOUT}\left(\mathbf{X}^{\ell+1}\right))$.

Not only do hidden layer anchors aggregate structural information over r hops, they induce a GNN that is now partially stochastic, as layers $1 \ldots r$ are deterministic. Indeed, by reducing network stochasticity, it is naturally expected that hidden layer anchoring will reduce the diversity of the hypotheses, but by sampling more *functionally diverse* hypotheses through deeper, semantically expressive anchors, it is possible that *naively* maximizing diversity is in fact not required for reliable uncertainty estimation. To validate this hypothesis, we thus propose the final variant, `READOUT` anchoring for graph classification tasks. While conceptually similar to hidden layer anchoring, here, we simultaneously minimize GNN stochasticity (only the classifier is stochastic) and maximize anchor expressivity (anchors are graph representations pooled after ℓ rounds of message passing). Notably, `READOUT` anchoring is also compatible with pretrained GNN backbones, as the final `MLP` layer of a pretrained model is discarded (if necessary), and reinitialized to accommodate query/anchor pairs. Given the frozen MPNNbackbone, only the anchored classifier head is trained.

In Sec. 5.5, we empirically verify the effectiveness of our proposed G-ΔUQ variants and demonstrate that fully stochastic GNNs are, in fact, unnecessary to obtain highly generalizable solutions, meaningful uncertainties and improved calibration on graph classification tasks.

5.4 Node Classification Experiments: G-ΔUQ Improves Calibration

In this section, we demonstrate that G-ΔUQ improves uncertainty estimation in GNNs, particularly when evaluating *node classifiers* under distribution shifts. To the best of our knowledge, GNN calibration has not been extensively evaluated under this challenging setting, where uncertainty estimates are known to be unreliable [169]. We demonstrate that G-ΔUQ not only directly provides better estimates, but also that combining G-ΔUQ with existing post-hoc calibration methods further improves performance.

Experimental Setup. We use the concept and covariate shifts for WebKB, Cora and CBAS datasets provided by [35], and follow the recommended hyperparameters for training. In our implementation of node feature anchoring, we use 10 random anchors to obtain predictions with G-ΔUQ. All our results are averaged over 5 seeds and post-hoc calibration methods (described further in App. C.9) are fitted on the in-distribution validation dataset. The expected calibration error and accuracy on the unobserved "OOD test" split are reported.

Results. From Table 5.1 (and expanded in Table. C.9), we observe that across 4 datasets and 2 shifts that G-ΔUQ, *without* any post-hoc calibration (✕), is superior to the vanilla model on nearly every benchmark for better or same accuracy (8/8 benchmarks) and better calibration error (7/8), often with a significant gain in calibration performance. Moreover, we note that combining G-ΔUQ with a particular posthoc calibration method improves performance relative to using the same posthoc method with a vanilla model. Indeed, on WebKB, across 9 posthoc strategies, "G-ΔUQ +¡calibration method¿" improves or maintains the calibration performance of the corresponding "no G-ΔUQ +¡calibration method¿" in 7/9 (concept) and 6/9 (covariate) cases. (See App. C.8 for more discussion.) Overall, across post hoc methods and evaluation sets, G-ΔUQ variants are very performant achieving (best

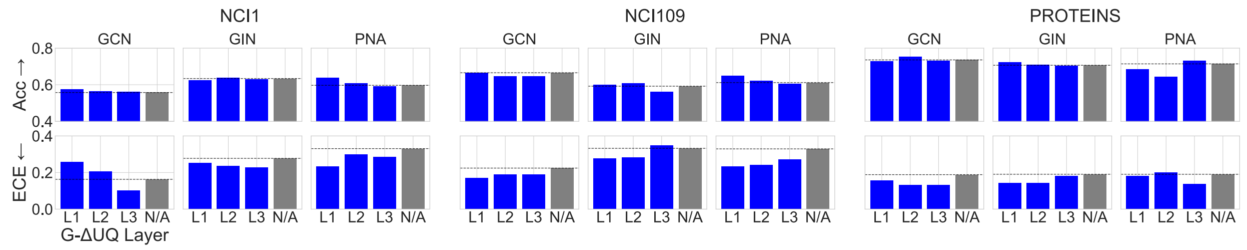


Figure 5.4: **Effect of Anchoring Layer.** Anchoring at different layers (L1, L2, L3) induces different hypotheses spaces. Variations of stochastic anchoring outperform models without it, and the lightweight `READOUT` anchoring in particular generally performs well across datasets and architectures.

accuracy: 8/8), best calibration (6/8) or second best calibration (2/8).

5.5 Graph Classification Uncertainty Experiments with G-ΔUQ

While applying G-ΔUQ to node classification tasks was relatively straightforward, performing stochastic centering with graph classification tasks is more nuanced. As discussed in Sec. 5.3, different anchoring strategies can introduce varying levels of stochasticity, and it is unknown how these strategies affect uncertainty estimate reliability. Therefore, we begin by demonstrating that fully stochastic GNNs are not necessary for producing reliable estimates (Sec. 5.5.1). We then extensively evaluate the calibration of partially stochastic GNNs on covariate and concept shifts with and without post-hoc calibration strategies (Sec. 5.5.2), as well as for different UQ tasks (Sec. 5.3). Lastly, we demonstrate that G-ΔUQ's uncertainty estimates remain reliable when used with different architectures and pretrained backbones (Sec. 5.6).

5.5.1 Is Full Stochasticity Necessary for G-ΔUQ?

By changing the anchoring strategy and intermediate anchoring layer, we can induce varying levels of stochasticity in the resulting GNNs. As discussed in Sec. 5.3, we hypothesize that the decreased stochasticity incurred by performing anchoring at deeper network layers will lead to more functionally diverse hypotheses, and consequently more reliable uncertainty estimates. We verify this hypothesis here, by studying the effect of anchoring layer on calibration under graph-size distribution shift. Namely, we find that `READOUT` anchoring sufficiently balances stochasticity and functional diversity.

Experimental Setup. We study the effect of different anchoring strategies on graph classification calibration under graph-size shift. Following the procedure of [17, 18], we create a size distribution shift by taking the smallest 50%-quantile of graph size for the training set, and evaluate on the largest 10% quantile. Following [17], we apply this splitting procedure to NCI1, NCI09, and PROTEINS [111], consider 3 GNN backbones (GCN [24], GIN [27], and PNA [26]) and use the same architectures/parameters. (See Appendix C.6 for dataset statistics.) The accuracy and expected calibration error over 10 seeds on the largest-graph test set are reported for models trained with and without stochastic anchoring.

Results. We compare the performance of anchoring at different layers in Fig. 5.4. While there is no clear winner across datasets and architectures for which *layer* to perform anchoring, we find there is consistent trend across all datasets and architectures the best accuracy and ECE is obtained by a G-ΔUQ variant. Overall, our results clearly indicate that partial stochasticity can yield substantial benefits when estimating uncertainty (though suboptimal layers selections are generally not too harmful). Insofar, as we are the first to focus on partially stochastic anchored GNNs, automatically selecting the anchoring layer is an interesting direction of future work. However, in subsequent experiments, we use `READOUT` anchoring, unless otherwise noted, as it is faster to train (see App. C.13), and allow our methods to support pretrained models. Indeed, `READOUT` anchoring (L3) yields top performance for some datasets and architectures such as PNA on PROTEINS, compared to earlier (L1, L2) and, as we discuss below, is very performative on a variety of tasks and shifts.

5.5.2 Calibration under Concept and Covariate Shifts

Next, we assess the ability of G-ΔUQ to produce well-calibrated models under covariate and concept shift in graph classification tasks. We find that G-ΔUQ not only provides better calibration out of the box, its performance is further improved when combined with post-hoc calibration techniques.

Experimental Setup. We use three different datasets (GOODCMNIST, GOODMotif-basis, GOODSST2) with their corresponding splits and shifts from the recently proposed Graph Out-Of Distribution (GOOD) benchmark [35]. The architectures and hyperparameters suggested by the benchmark are used for training. G-ΔUQ uses `READOUT` anchoring and 10 random anchors (see App. C.7 for more details). We report accuracy and expected calibration error for the OOD test dataset, taken over three seeds.

Results. As shown in Table 5.1, we observe that G-ΔUQ leads to inherently better calibrated models, as the ECE from G-ΔUQ without additional post-hoc calibration (✕)

Table 5.1: **Calibration under Covariate and Concept shifts.** G-ΔUQ leads to better calibrated models for node-(GOODCora) and graph-level prediction tasks under different kinds of distribution shifts. Notably, G-ΔUQ can be combined with post-hoc calibration techniques to further improve calibration. The expected calibration error (ECE) is reported. Best, Second.

			Shift: Concept				Shift: Covariate			
			Accuracy (↑)		ECE (↓)		Accuracy (↑)		ECE (↓)	
Dataset	Domain	Calibration	No G-Δ UQ	G-Δ UQ	No G-Δ UQ	G-Δ UQ	No G-Δ UQ	G-Δ UQ	No G-Δ UQ	G-Δ UQ
GOODCora	Degree	×	0.581±0.003	0.595±0.003	0.307±0.009	0.13±0.011	0.47±0.002	0.518±0.014	0.348±0.032	0.141±0.008
		CAGCN	0.581±0.003	**0.597**±0.002	0.135±0.009	0.128±0.025	0.47±0.002	0.522±0.025	0.256±0.08	0.231±0.025
		Dirichlet	0.534±0.007	0.551±0.004	0.12±0.004	0.196±0.003	0.414±0.007	0.449±0.01	0.163±0.002	0.356±0.01
		ETS	0.581±0.003	0.596±0.004	0.301±0.009	0.116±0.018	0.47±0.002	**0.523**±0.003	0.31±0.077	0.141±0.003
		GATS	0.581±0.003	0.596±0.004	0.185±0.018	0.229±0.039	0.47±0.002	0.521±0.011	0.211±0.004	0.308±0.011
		IRM	0.582±0.002	0.597±0.002	0.125±0.001	0.102±0.002	0.469±0.001	0.522±0.004	0.194±0.005	0.13±0.004
		Orderinvariant	0.581±0.003	0.592±0.002	0.226±0.024	0.213±0.049	0.47±0.002	0.498±0.027	0.318±0.042	0.196±0.027
		Spline	0.571±0.003	0.595±0.003	0.080±0.004	**0.068**±0.004	0.459±0.003	0.52±0.004	0.158±0.01	**0.098**±0.004
		VS	0.581±0.003	0.596±0.004	0.306±0.004	0.127±0.002	0.47±0.001	0.522±0.005	0.345±0.005	0.146±0.005
GOODCMNIST	Color	×	0.499±0.003	0.497±0.002	0.439±0.078	0.334±0.066	0.348±0.009	0.355±0.034	0.551±0.147	0.423±0.172
		Dirichlet	0.495±0.009	0.510±0.008	0.303±0.012	0.304±0.007	0.350±0.053	0.335±0.059	0.542±0.091	0.406±0.076
		ETS	0.499±0.011	0.500±0.013	0.433±0.014	0.359±0.013	0.348±0.037	0.336±0.067	0.538±0.077	0.467±0.088
		IRM	0.499±0.006	0.500±0.010	0.285±0.004	**0.283**±0.008	0.348±0.049	0.336±0.071	0.416±0.084	0.425±0.093
		Orderinvariant	0.499±0.030	0.500±0.028	0.379±0.050	0.386±0.042	0.348±0.036	0.337±0.059	0.475±0.077	0.542±0.104
		Spline	0.495±0.008	0.497±0.010	0.29±0.007	0.291±0.008	0.346±0.051	0.335±0.071	0.414±0.085	0.425±0.093
		VS	0.499±0.007	0.500±0.012	0.439±0.006	0.377±0.009	0.349±0.037	0.336±0.067	0.549±0.071	0.468±0.089
		Ensembling	0.505±0.001	**0.509**±0.004	0.437±0.082	0.343±0.004	0.397±0.005	**0.408**±0.006	0.423±0.017	**0.327**±0.013
GOODMotif	Basis	×	0.925±0.001	0.925±0.003	0.095±0.014	0.078±0.007	0.691±0.001	0.689±0.002	0.329±0.274	0.342±0.266
		Dirichlet	0.925±0.011	0.923±0.010	0.081±0.015	0.103±0.007	0.686±0.009	0.681±0.009	0.337±0.067	0.316±0.047
		ETS	0.925±0.009	0.927±0.012	0.095±0.010	0.096±0.013	0.691±0.011	0.699±0.016	0.314±0.041	0.304±0.049
		IRM	0.925±0.014	0.93±0.013	0.087±0.018	0.097±0.010	0.691±0.011	0.698±0.016	0.316±0.051	0.305±0.045
		Orderinvariant	0.925±0.010	0.928±0.011	0.091±0.009	0.093±0.007	0.691±0.011	0.690±0.011	0.321±0.050	0.319±0.041
		Spline	0.925±0.010	0.927±0.011	0.091±0.008	0.089±0.012	0.691±0.010	0.689±0.016	0.324±0.055	0.313±0.051
		VS	0.925±0.009	0.927±0.012	0.095±0.010	0.095±0.013	0.683±0.013	0.680±0.018	0.326±0.057	0.311±0.059
		Ensembling	0.932±0.002	**0.943**±0.006	0.086±0.016	**0.047**±0.003	**0.714**±0.012	0.699±0.009	**0.298**±0.383	0.321±0.196
GOODSST2	Length	×	0.694±0.002	0.693±0.001	0.288±0.017	0.277±0.011	0.826±0.002	0.828±0.004	0.159±0.027	0.154±0.039
		Dirichlet	0.686±0.02	0.683±0.001	0.15±0.021	**0.138**±0.015	0.793±0.005	0.8±0.012	0.15±0.02	**0.131**±0.007
		ETS	0.685±0.02	0.683±0.001	0.21±0.009	0.211±0.003	0.794±0.005	0.8±0.011	0.287±0.007	0.296±0.014
		IRM	0.685±0.019	0.682±0.002	0.239±0.002	0.231±0.006	0.796±0.006	0.801±0.011	0.26±0.005	0.265±0.011
		Orderinvariant	0.685±0.02	0.683±0.001	0.225±0.002	0.222±0.003	0.794±0.005	0.8±0.011	0.226±0.003	0.224±0.007
		Spline	0.684±0.02	0.683±0.002	0.233±0.005	0.23±0.005	0.79±0.004	0.794±0.016	0.259±0.005	0.263±0.012
		VS	0.685±0.019	0.683±0	0.334±0.044	0.374±0.002	0.787±0.008	0.8±0.013	0.307±0.116	0.32±0.011
		Ensembling	0.705±0.002	**0.709**±0.004	0.276±0.038	0.248±0.022	0.838±0.001	**0.842**±0.006	0.154±0.032	0.132±0.019

is better than the vanilla ("No G-ΔUQ") counterparts on 5/6 datasets. Moreover, we find that combining G-ΔUQ with a particular post-hoc calibration methods further elevates its performance relative to combining the same strategy with vanilla models. Indeed, for a fixed post-hoc calibration strategy, G-ΔUQ improves the calibration, while maintaining comparable if not better accuracy on the vast majority of the methods and datasets. There are some settings where combining G-ΔUQ or the vanilla model with a post-hoc method leads decreases performance (for example, GOODSST2, covariate, ETS, calibration) but we emphasize that this is not a short-coming of G-ΔUQ. Posthoc strategies, which rely upon ID calibration datasets, may not be effective on shifted data. This further emphasizes the importance of our OOD evaluation and G-ΔUQ as an intrinsic method for improving uncertainty estimation.

Table 5.2: **GOOD-Datasets, OOD Detection Performance.** The AUROC of the binary classification task of classifying OOD samples is reported. G-ΔUQ variants outperform the vanilla models on 6/8 datasets. We further note that end-to-end G-ΔUQ does in fact lose performance relative to the vanilla model on 4 datasets. Investigating why pretrained G-ΔUQ is able to increase performance on those datasets is an interesting direction of future work. It does not appear that a particular shift is more difficult for this task: concept shift is easier for GOODCMNIST and GOODMotif(Basis) while covariate shift is easier for GOODMotif(Size) and GOODSST2. Combining G-ΔUQ with more sophisticated, uncertainty or confidence based OOD scores may further improve performance.

	CMNIST (Color)		**MotifLPE (Basis)**		**MotifLPE (Size)**		**SST2**	
Method	**Concept(↑)**	**Covariate(↑)**	**Concept(↑)**	**Covariate(↑)**	**Concept(↑)**	**Covariate(↑)**	**Concept(↑)**	**Covariate(↑)**
Vanilla	0.759 ± 0.006	0.468 ± 0.092	0.736 ± 0.021	$\mathbf{0.466 \pm 0.001}$	0.680 ± 0.003	0.755 ± 0.074	$\mathbf{0.350 \pm 0.014}$	0.345 ± 0.066
G-ΔUQ	0.771 ± 0.002	0.470 ± 0.043	0.758 ± 0.006	0.328 ± 0.022	0.677 ± 0.005	0.691 ± 0.067	0.338 ± 0.023	0.351 ± 0.042
Pretr. G-ΔUQ	$\mathbf{0.774 \pm 0.016}$	$\mathbf{0.543 \pm 0.152}$	$\mathbf{0.769 \pm 0.029}$	0.272 ± 0.025	$\mathbf{0.686 \pm 0.004}$	$\mathbf{0.829 \pm 0.113}$	0.324 ± 0.055	$\mathbf{0.446 \pm 0.049}$

Table 5.3: **RotMNIST-Calibration.** Here, we report expanded results (calibration) on the Rotated MNIST dataset, including a variant that combines G-ΔUQ with Deep Ens. Notably, we see that anchored ensembles outperform basic ensembles in both accuracy and calibration.

Architecture	LPE?	G-ΔUQ	Calibration	Avg.ECE (↓)	ECE (10) (↓)	ECE (15) (↓)	ECE (25) (↓)	ECE (35) (↓)	ECE (40) (↓)
GatedGCN	×	×	×	0.038 ±0.001	0.059 ±0.001	0.068 ±0.340	0.126 ±0.008	0.195 ±0.012	0.245 ±0.011
	×	✓	×	**0.018** ±0.008	0.029 ±0.013	**0.033** ±0.164	0.069 ±0.033	0.117 ±0.048	0.162 ±0.067
	×	×	Ensembling	0.026 ±0.000	0.038 ±0.001	0.042 ±0.001	0.084 ±0.002	0.135 ±0.001	0.185 ±0.003
	×	✓	Ensembling	**0.014** ±0.003	**0.018** ±0.005	**0.021** ±0.005	0.036 ±0.012	0.069 ±0.032	0.114 ±0.056
GatedGCN	✓	×	×	0.036 ±0.003	0.059 ±0.002	0.068 ±0.340	0.125 ±0.006	0.191 ±0.007	0.240 ±0.008
	✓	✓	×	0.022 ±0.007	**0.028** ±0.014	0.034 ±0.169	**0.062** ±0.022	**0.109** ±0.019	**0.141** ±0.019
	✓	×	Ensembling	0.024 ±0.001	0.038 ±0.001	0.043 ±0.002	0.083 ±0.001	0.139 ±0.004	0.181 ±0.002
	✓	✓	Ensembling	0.017 ±0.002	0.024 ±0.005	0.027 ±0.008	**0.030** ±0.004	**0.036** ±0.012	**0.059** ±0.033
GPS	✓	×	×	0.026 ±0.001	0.044 ±0.001	0.052 ±0.156	0.108 ±0.006	0.197 ±0.012	0.273 ±0.008
	✓	✓	×	0.022 ±0.001	0.037 ±0.005	0.044 ±0.133	0.091 ±0.008	0.165 ±0.018	0.239 ±0.018
	✓	×	Ensembling	0.016 ±0.001	0.026 ±0.002	0.030 ±0.000	0.066 ±0.000	0.123 ±0.000	0.195 ±0.000
	✓	✓	Ensembling	**0.014** ±0.000	0.023 ±0.002	0.027 ±0.003	0.055 ±0.004	0.103 ±0.006	0.164 ±0.006

5.5.3 Using Confidence Estimates in Safety-Critical Tasks

While post-hoc calibration strategies rely upon an additional calibration dataset to provide meaningful uncertainty estimates, such calibration datasets are not always available and may not necessarily improve OOD performance [169]. Thus, we also evaluate the quality of the uncertainty estimates directly provided by G-ΔUQ on two additional UQ-based, safety-critical tasks [166, 32, 167]: (i) OOD detection [174], which attempts to classify samples as in- or out-of-distribution, and (ii) generalization error prediction (GEP) [61], which attempts to predict the generalization on unlabeled test datasets (to the best of our knowledge, we are the first to study GEP of graph classifiers). In the interest of space, we present the results on GEP in the appendix.

OOD Detection Experimental Setup. By reliably detecting OOD samples and abstaining from making predictions on them, models can avoid over-extrapolating to irrelevant distributions. While many scores have been proposed for detection [174, 187, 188, 189, 190], popular scores, such as maximum softmax probability and predictive entropy [62], are derived from uncertainty estimates. Here, we report the AUROC for the binary classification task of detecting OOD samples using the maximum softmax probability as the score [191].

OOD Detection Results. As shown in Table 5.2, we observe that G-ΔUQ variants improve OOD detection performance over the vanilla baseline on 6/8 datasets, where pretrained G-ΔUQ obtains the best overall performance on 6/8 datasets. G-ΔUQ performs comparably on GOODSST2(concept shift), but does lose some performance on GOODMotif(Covariate). We note that vanilla models provided by the original benchmark generalized poorly on this particular dataset (increased training time/accuracy did not improve performance), and this behavior was reflected in our experiments. We suspect that poor generalization coupled with stochasticity may explain G-ΔUQ's performance here.

5.6 Fine Grained Analysis of G-ΔUQ

Given that the previous sections extensively verified the effectiveness of G-ΔUQ on a variety of covariate and concept shifts across several tasks, we seek a more fine-grained understanding of G-ΔUQ's behavior with respect to different architectures and training strategies. In particular, we demonstrate that G-ΔUQ continues to improve calibration with expressive graph transformer architectures, and that using `READOUT` anchoring with pretrained GNNs is an effective lightweight strategy for improving calibration of frozen GNN models.

5.6.1 Calibration under Controlled Shifts

Recently, it was shown that modern, non-convolutional architectures [192] are not only more performant but also more calibrated than older, convolutional architectures [34] under vision distribution shifts. Here, we study an analogous question: are more expressive GNN architectures better calibrated under distribution shift, and how does G-ΔUQ impact their calibration? Surprisingly, we find that more expressive architectures are not considerably better calibrated than their MPNN counterparts, and ensembles of MPNNs outperform ensembles of GTrans. Notably, G-ΔUQ continues to improve calibration with respect to these architectures as well.

Experimental Setup. *(1) Models.* While improving the expressivity of GNNs is an active area of research, positional encodings (PEs) and graph-transformer (GTran) architectures [193] are popular strategies due to their effectiveness and flexibility. GTrans not only help mitigate over-smoothing and over-squashing [194, 195] but they also better capture long-range dependencies [196].

Meanwhile, graph PEs help improve expressivity by differentiating isomorphic nodes, and capturing structural vs. proximity information [186]. Here, we ask if these enhancements translate to improved calibration under distribution shift by comparing architectures with/without PEs and transformer vs. MPNN models. We use equivariant and stable PEs [185], the state-of-the-art, "general, powerful, scalable" (GPS) framework with a GatedGCN backbone for the GTran, GatedGCN for the vanilla MPNN, and perform `READOUT` anchoring with 10 random anchors. *(2) Data.* In order to understand calibration behavior as distribution shifts become progressively more severe, we create structurally distorted but valid graphs by rotating MNIST images by a fixed number of degrees [12] and then creating the corresponding super-pixel graphs [86, 147, 25]. (See Appendix, Fig. C.2.) Since superpixel segmentation on these rotated images will yield different superpixel k-nn graphs but leave class information unharmed, we can emulate different severities of label-preserving structural distortion shifts. We note that models are trained only using the original ($0°$ rotation) graphs. Accuracy (see appendix) and ECE over 3 seeds are reported for the rotated graphs.

Results. In Table 5.3, we present the OOD calibration results, with results of more variants and metrics in the supplementary Table C.2 and C.5. First, we observe that PEs have minimal effects on both calibration and accuracy by comparing GatedGCN with and without LPEs. This suggests that while PEs may enhance expressivity, they do not directly induce better calibration. Next, we find that while vanilla GPS is better calibrated when the distribution shift is not severe (10, 15, 25 degrees), it is less calibrated (but more performant) than GatedGCN at more severe distribution shifts (35, 40 degrees). This is in contrast to known findings about vision transformers. Lastly, we see that G-ΔUQ continues to improve calibration across all considered architectural variants, with minimal accuracy loss. *Surprisingly, however, we observe that ensembles of G-ΔUQ models not only effectively resolve any performance*

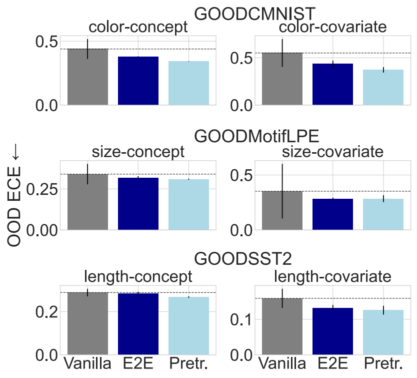


Figure 5.5: **Out-of-distribution Calibration Error.** G-ΔUQ is applied in end-to-end training vs. to a pretrained model, which is a simple yet effective way to use stochastic anchoring.

drops, they also cause MPNNs to be better calibrated than their GTran counterparts.

5.6.2 How does G-ΔUQ perform with pretrained models?

As large-scale pretrained models become the norm, it is beneficial to be able to perform lightweight training that leads to safer models. Thus, we investigate if `READOUT` anchoring is such a viable strategy when working with pretrained GNN backbones, as it only requires training a stochastically centered classifier on top of a frozen backbone. (Below, we discuss results on GOODDataset, but please see C.4 for results on RotMNIST and C.12 for additional discussion.)

Results. From Fig. 5.5 (and expanded in Fig. C.4), we observe that across datasets, pretraining (PT) yields competitive (often superior) OOD calibration with respect to end-to-end (E2E) G-ΔUQ. With the exception of GOODMotif (basis) dataset, PT G-ΔUQ improves the OOD ECE over both vanilla and E2E G-ΔUQ models at comparable or improved OOD accuracy (6/8 datasets). Furthermore, PT G-ΔUQ also improves the ID ECE on all but the GOODMotif(size) (6/8), where it performs comparably to the vanilla model, and maintains the ID accuracy. Notably, as only an anchored classifier is trained, PT G-ΔUQ substantially reduces training time relative to E2E G-ΔUQ and vanilla models (see App. C.13), highlighting its strengths as a light-weight, effective strategy for improving uncertainty estimation.

5.7 Conclusion

We propose G-ΔUQ, a novel training approach that adapts stochastic data centering for GNNs through newly introduced graph-specific anchoring strategies. Our extensive experiments demonstrate G-ΔUQ improves calibration and uncertainty estimates of GNNs under distribution shifts.

CHAPTER 6

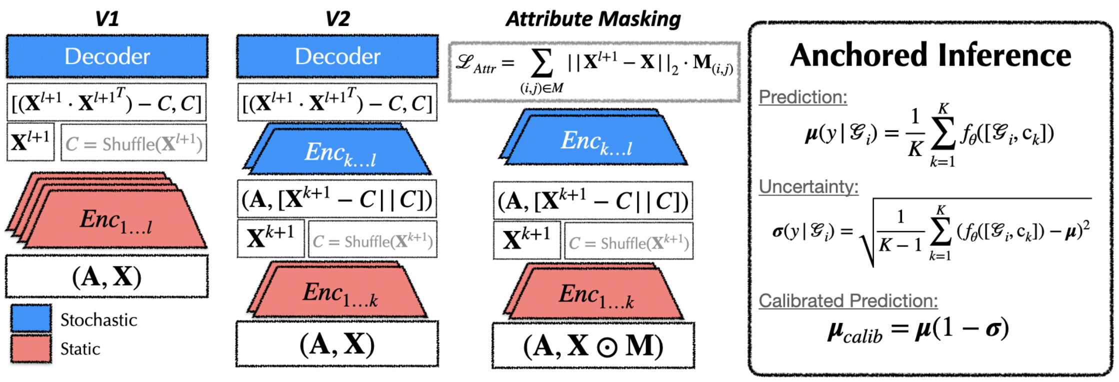
On Link Prediction Calibration with Stochastic Centering

6.1 Introduction

Having discussed improving the uncertainty estimates of GNN-based node and graph classifiers in the preceding chapter, here, we focus on link prediction (LP) [197, 22], which as high-impact applications ranging from product recommendation [23] to biological network completion (e.g., gene-gene interaction or drug-drug interactions) [20, 96]. Most often, the predicted links are used to invoke expensive actions or time-consuming experiments. Consequently, in addition to obtaining accurate predictions, it is important that practitioners are able to trust a model's confidence in its predictions [32]. This has led to the emergence of a large class of calibration techniques [179, 34, 198, 176]. Calibration is the process of adjusting output probabilities or confidence scores produced by a model to ensure that they accurately reflect the true likelihood associated with a specific prediction [199, 34]. While most existing studies on GNN calibration have extensively focused on node [42, 41, 200] or graph classification [81], the calibration behavior of LP models remains considerably less studied.

At a high level, link prediction architectures contain an encoder, which produces node-level features, and a light-weight decoder, which aggregates two node representations $(v_{\mathrm{i}}, v_{\mathrm{j}})$ to predict whether a given edge $(e_{(i,j)})$ is plausible. Indeed, it is challenging to directly extend state-of-the-art approaches for improving calibration from node or graph classification literature to LP settings. Though one can systematically estimate uncertainties from the encoder module, the lack of any node-level task makes it challenging to ensure that those

The material in this chapter is derived from the paper "On Estimating Link Prediction Uncertainty using Stochastic Centering" [79], which appeared in the proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing in 2024. Code can be accessed here.



V1
V2
Attribute Masking
Anchored Inference
Decoder
$[(\mathbf{X}^{l+1} \cdot \mathbf{X}^{l+1^T}) - C, C]$
$\mathbf{X}^{l+1}$
$C = \text{Shuffle}(\mathbf{X}^{l+1})$
$Enc_{1...l}$
$(\mathbf{A}, \mathbf{X})$
Stochastic
Static
$Enc_{k...l}$
$(\mathbf{A}, [\mathbf{X}^{k+1} - C || C])$
$\mathbf{X}^{k+1}$
$C = \text{Shuffle}(\mathbf{X}^{k+1})$
$Enc_{1...k}$
$\mathscr{L}_{Attr} = \sum_{(i,j) \in M} ||\mathbf{X}^{l+1} - \mathbf{X}||_2 \cdot \mathbf{M}_{(i,j)}$
$(\mathbf{A}, \mathbf{X} \odot \mathbf{M})$
Prediction:
$\mu(y | \mathscr{G}_i) = \frac{1}{K} \sum_{k=1}^{K} f_\theta([\mathscr{G}_i, c_k])$
Uncertainty:
$\sigma(y | \mathscr{G}_i) = \sqrt{\frac{1}{K-1} \sum_{k=1}^{K} (f_\theta([\mathscr{G}_i, c_k]) - \mu)^2}$
Calibrated Prediction:
$\mu_{calib} = \mu(1 - \sigma)$

Figure 6.1: **Overview of E-ΔUQ.** We propose three different stochastic centering variants that induce varying levels of stochasticity in the underlying GNN. Variant (**E-ΔUQ (v1)**) directly models the epistemic uncertainties arising from the sampling of edges in different parts of the (node) feature space. Variant (**E-ΔUQ (v2)**) performs stochastic centering in the encoder network itself and implicitly leverage those uncertainties to produce calibrated edge probabilities. **E-ΔUQ (v3)**) uses the auxiliary attribute masking task to first calibrate the node-level uncertainties and subsequently estimate the edge-level uncertainties similar to **E-ΔUQ (v2)**. We show the attribute masking task above and use shuffled node features as the anchoring distribution.

uncertainties are well calibrated. Furthermore, it is highly non-trivial to model the interaction between node uncertainties for different LP decoder choices (e.g., node feature concatenation, scalar dot product). For example, it is unclear if an edge (v_i, v_j) between two nodes with high (node-level) uncertainties in their features is always guaranteed to have higher (edge-level) uncertainty compared to another edge $(v_\text{i}', v_\text{j}')$ with only one node with high uncertainty.

Recently, [201] proposed a Bayesian approach for link prediction, which places an explicit prior over node features (in each layer) and uses a hierarchical Gaussian process (GP) to combine node-level priors to obtain edge-level predictions. While such an approach allows for closed-form aggregation of node-level uncertainties, it has a number of challenges. First, it requires a specific link-predictor structure (hierarchical edge-GP), which may not be compatible with the different decoder choices used in practice. Second, given the high computational costs associated with GP inferencing, this can be especially problematic when scaling to larger, production-scale datasets. Third, it is not straightforward to integrate any additional (or auxiliary) node-level tasks that can help better calibrate the node features [95].

To circumvent these challenges, we propose a non-parametric, architecture-agnostic, LP uncertainty estimator based on the recently proposed stochastic centering framework [173, 182]. We choose this framework for its flexibility to be adopted to any architecture, as well as, its strong generalization behavior under challenging distribution shifts [81].

Extending stochastic centering to edge-level uncertainty (Sec. 3): We first extend stochastic centering to link prediction networks by considering the node features to be deterministic and enabling uncertainty estimation only in the decoder module. This variant (**E-ΔUQ (v1)**) directly models the epistemic uncertainties arising from the sampling of edges in different parts of the (node) feature space.

Creating Meaningful Node-level uncertainties (Sec. 3): Despite the simplicity of the previous variant and its performance in practice, incorporating the node-level uncertainties can lead to richer LP models. Hence, we propose to invoke stochastic centering in the encoder network itself (**E-ΔUQ (v2)**) and implicitly leverage those uncertainties to produce calibrated edge probabilities. Finally, we consider a sophisticated variant (**E-ΔUQ (v3)**) where we leverage an auxiliary task to first calibrate the node-level uncertainties and subsequently estimate the edge-level uncertainties similar to **E-ΔUQ (v2)**.

Experimental Evaluation (Sec. 4): Using a suite of citation network datasets, we systematically evaluate the three proposed uncertainty estimation techniques and demonstrate their behavior in terms of both fidelity of the predicted links and calibration error metrics. This work, for the first time, delves into the important problem of appropriately handling node-level uncertainties in LP architectures.

6.2 Background & Related Work

In this section, we briefly discuss the notations and related work relevant to problem setting and approach.

Notations. Let $\mathcal{G} = (\mathbf{X}, \mathbf{A})$ be a graph with node features $\mathbf{X} \in \mathbb{R}^{N \times d_\ell}$, and adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where N, m, d_ℓ, denotes the number of nodes, number of edges, and feature dimension. We wish to predict unobserved edges that are missing from $\mathbf{A}$ given the observed, training graph and nodes. Thus, we can define a LP GNN consisting of a node `Encoder` with ℓ message passing layers ($MPNN$), and `Decoder` which predicts whether an edge $e(i, j)$ exists between two nodes, given their representations (X_i, X_j):

$$\mathbf{X}^{\ell+1} = \texttt{Encoder}\left(\mathbf{X}^{\ell}, \mathbf{A}\right), \tag{6.1}$$

$$E_{(i,j)} = \texttt{Decoder}\left(\mathbf{X}_i^{\ell+1}\mathbf{X}_j^{\ell+1}\right) \tag{6.2}$$

where $\mathbf{X}^{\ell+1}$ is the intermediate node representations. Popular decoder architectures include taking the dot product or concatenating representations and then passing the resulting representation through some linear layers. Models are trained by treating LP as a binary classification task, where true edges in $\mathbf{A}$ are considered positive class samples and non-edges in $\mathbf{A}$ are considered negative class samples.

Calibration and Stochastic Centering. While several strategies have been proposed to improve calibration [179, 34, 198, 176] of vision models and GNN-based node classification calibration [42, 41, 200, 81], these methods are not suited for link prediction calibration because they cannot ensure reliable node-level calibration without node-level supervision (see Sec. 6.1) and often struggle to outperform the simple, but prohibitively expensive deep ensemble (DEns) [179] baseline. Unfortunately, DEns, which takes the mean prediction over a set of independently trained models, requires training and storing multiple models. Recently, however, [173] proposed a state-of-the-art, single model uncertainty estimation method, Δ-UQ, based on the principle of *anchoring*, which is capable of simulating the behavior of an ensemble through only a single model.

Conceptually, anchoring is the process of creating a relative representation for an input sample x in terms of a random anchor c (which is used to perform the *stochastic centering*), $[x - c, c]$. By choosing different anchors randomly in each training iteration, Δ-UQ emulates the process of sampling different solutions from the hypothesis space (akin to an ensemble). During inference, Δ-UQ aggregates multiple predictions obtained via different random anchors and produces uncertainty estimates.

Formally, given a trained stochastically centered model, $f_\theta : [\mathbf{X} - \mathbf{C}, \mathbf{C}] \rightarrow \hat{\mathbf{Y}}$, let $\mathbf{C} := \mathbf{X}_{train}$ be the anchor distribution, $x \in \mathbf{X}_{test}$ be a test sample, and anchor $c \in \mathbf{C}$ be anchor. Then, the mean target class prediction, $\boldsymbol{\mu}(y|\mathrm{x})$, and corresponding variance, $\boldsymbol{\sigma}(y|\mathrm{x})$ over K random anchors are computed as:

$$\boldsymbol{\mu}(y|\mathrm{x}) = \frac{1}{K}\sum_{k=1}^{K} f_\theta([\mathrm{x} - \mathrm{c}_k, \mathrm{c}_k]) \tag{6.3}$$

$$\boldsymbol{\sigma}(y|\mathrm{x}) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}(f_\theta([\mathrm{x} - \mathrm{c}_k, \mathrm{c}_k]) - \boldsymbol{\mu})^2} \tag{6.4}$$

Since the variance over K anchors captures epistemic uncertainty by sampling different hypotheses, these estimates can be used to modulate the predictions: $\boldsymbol{\mu}_{\text{calib.}} = \boldsymbol{\mu}(1 - \boldsymbol{\sigma})$. The resulting calibrated predictions and uncertainty estimates have led to state-of-the-art performance on vision [173, 182] and graph classification tasks [81], while still only requiring a single model. Given its impressive performance and flexibility, we focus on adapting stochastic centering to our link prediction calibration setting. Namely, we discuss in detail design considerations that arise from aggregating node-level uncertainties into edge-level confidence estimates.

6.3 Proposed Approach

In this section, we introduce our proposed approach, and discuss the three variants, for improving GNN-based link prediction calibration using stochastic centering (see Fig. 6.1). Furthermore, we will demonstrate the importance of creating and aggregating meaningful node-level uncertainties. Indeed, as discussed in Sec. 6.1, on the one hand, LP calibration is difficult since node-level uncertainties must be correctly aggregated in order to produce consistent edge-level uncertainties. On the other hand, due to the lack of suitable node-level tasks, the uncertainties associated with the node features can themselves be poorly calibrated. Consequently, even a sophisticated aggregation strategy in LP decoders can lead to sub-par calibration of edge probabilities. Below, we introduce three different variants of our proposed method, where each variant gradually seeks to improve the characterization of node-level uncertainties, and subsequently leverages the stochastic centering framework to produce edge uncertainties.

E-ΔUQ (v1) – Deterministic node features: While stochastic centering can be performed at any layer of the GNN, in this variant, stochastic centering is only performed prior to decoder. In other words, the node representations are assumed to be deterministic and that the epistemic uncertainties arise from the non-uniform samples of edges in different parts of the feature space. While, this assumption indicates that we expect the least amount of change on the reliability of node-level features, we perform stochastic feature aggregation over pairs of nodes by utilizing the anchoring framework. Given that the LP decoder paramterizes an edge based on a chosen aggregation function on the given pair of nodes, through anchoring, we sample different possible aggregation hypotheses before marginalizing over anchors according to Eq. 6.6. In other words, given the features $\mathbf{x}_i$ and $\mathbf{x}_j$ for a node pair, we perform stochastic centering using a randomly chosen anchor $\boldsymbol{c}$ (from a pre-specified anchor distribution). Formally, we define this operation for dot-product and concatenation-style LP decoder modules as follows:

$$\texttt{Decoder}_{dot} : [(\mathbf{x}_i - \mathbf{c}) * (\mathbf{x}_j - \mathbf{c}), \mathbf{c}] \tag{6.5}$$

$$\texttt{Decoder}_{concat} : [(\mathbf{x}_i - \mathbf{c}||\mathbf{x}_j - \mathbf{c}), \mathbf{c}] \tag{6.6}$$

Similar to standard anchored model training, this tuple is taken as input by the LP decoder and trained using the standard cross entropy loss. Note, in each iteration of training, a different anchor $\mathbf{c}$ is randomly chosen.

E-ΔUQ (v2) – Partially stochastic encoder: As discussed earlier, it is reasonable to expect the node-level uncertainties can be utilized to improve the calibration of the edge probabilities obtained using LP. This is motivated by the fact that, the encoder architecture (implemented using GNNs) can be susceptible to epistemic uncertainties arising from the distribution of node attributes and hence, they can singificantly influence the subsequent predictions on pairs of nodes. Hence, in this variant, we invoke anchoring in the intermediate layers of the encoder architecture itself, and obtain stochastic node representations. Formally,

$$\begin{aligned}
\mathbf{X}^{r+1} &= \texttt{Encoder}^{1\ldots r}(\mathbf{X}, \mathbf{A}) \\
\mathbf{X}^{\ell+1} &= \texttt{Encoder}^{r+1\ldots\ell}\left([\mathbf{X}^{r+1} - \mathbf{C}, \mathbf{C}], \mathbf{A}\right) \\
\hat{E}_{(i,j)} &= \texttt{Decoder}\left(\mathbf{X}_i^{\ell+1}, \mathbf{X}_j^{\ell+1}\right)
\end{aligned}$$

However, it is important to note that this approach results in partially stochastic encoder model (i.e., first few layers of the encoder are deterministic) and the anchoring process leverages the structural information (through the message passing in GNN layers). By introducing anchoring in the encoder and performing end-to-end training of the LP model, we are able to effectively sample the hypothesis space for joint node feature learning and LP decoding. We expect this increased diversity to help improve the quality of our link-level predictions.

E-ΔUQ (v3) – Partially stochastic encoders + Node-level pretraining. While the aforementioned variants use stochastic centering to implicitly improve the aggregation of uncertainty over pairs of nodes, they do not explicitly improve the quality of node-level uncertainties. Indeed, this is difficult as node-level calibration supervision cannot be assumed on LP tasks. Therefore, we combine E-ΔUQ (v2) with an unsupervised node-level pretraining task to prime the encoder's node-level uncertainties before LP training. In particular, we pretrain the encoder with an auxillary node feature attribute masking task, and then train both the encoder and decoder with the standard link-prediction loss. Formally, assuming $\mathbf{M} \in [0,1]^{N\times d}$ denotes a random binary mask, we use that to mask portions of the input node feature matrix and define a self-supervised objective as follows:

$$\begin{aligned}
\mathbf{X}^{r+1} &= \texttt{Encoder}^{1\ldots r}(\mathbf{X} \odot \mathbf{M}, , \mathbf{A}) \\
\mathbf{X}^{\ell+1} &= \texttt{Encoder}^{r+1\ldots\ell}\left([\mathbf{X}^{r+1} - \mathbf{C}, \mathbf{C}], \mathbf{A}\right) \\
\mathcal{L}_{Attr} &= \sum_{(i,j)\in\mathbf{M}} ||\mathbf{X}^{\ell+1} - \mathbf{X}||_2 \cdot \mathbf{M}_{(i,j)}
\end{aligned}$$

Here, $\|.\|_2$ denotes the ℓ_2 norm and the reconstruction loss is measures only using the masked parts of the feature matrix. After completing the attribute masking-based pretraining, the encoder is equipped to produce node-level uncertainties. Subsequently, both the encoder and decoder modules of the LP architecture are trained end-to-end, following E-ΔUQ (v2). While other pretraining tasks can be considered, we use attribute masking for effectiveness and ease of implementation.

6.4 Experiments

In this section, we experimentally validate the effectiveness of our three LP calibration variants. *Experimental Set-up:* We consider three different datasets (Cora, Citeseer, Pubmed) and use the publicly available train-test splits for evaluation. A 3-layer GraphSAGE [28] backbone is used for the encoder, with either a dot-product decoder (Cora) or a concatenation decoder (Citeseer, Pubmed) (Table 6.4). Ten anchors are used for all E-ΔUQ variants. Hyper-parameters are shared between vanilla and E-ΔUQ models. The AUPRC and expected calibration error are reported over 10 seeds. We report results for the best intermediate anchoring layer according to validation AUPRC. We make the following observations from Table 6.4.

Table 6.1: **Dataset Statistics.**

Name	#nodes	#edges	#features
Cora	2,708	10,556	1,433
CiteSeer	3,327	9,104	3,703
PubMed	19,717	88,648	500

Observation 1: Stochastic centering variants improves the calibration on all datasets over the vanilla model. Indeed, the improvement is particularly large on Cora, where ECE is decreased by 50%, and Citeseer, where ECE is decreased by 16%. While we do not see as large gains on Pubmed, we do note that no E-ΔUQ variants increases the calibration error. This clearly suggests that our stochastic centering approach is effective.

Observation 2: Stochastic centering variants perform comparably on AUPR, with E-ΔUQ variants performing the best on 2/3 datasets. Generally, we see that E-ΔUQ variants improve AUPR 4/9, though we suspect that E-ΔUQ performance could be further improved if we tuned method-specific hyper-parameters.

Observation 3: Amongst E-ΔUQ variants, E-ΔUQ (v3) obtains the best calibration on 2/3 datasets. This suggests there is value to our pretraining method, which seeks to improve node-level calibration. We suspect that E-ΔUQ (v3)'s performance could be further improved with better auxiliary tasks, but we leave the design of such tasks to future work.

Observation 4: E-ΔUQ (v2) induced better calibration than E-ΔUQ (v1) on 3/3 datasets, and has better AUPR on 2/3 datasets. This supports our argument better node-level calibration is critical for also improving LP calibration.

6.5 Conclusion

In this work, we proposed and evaluated three variants of stochastic centering for improving the calibration of graph neural networks for link prediction. Our key finding is that properly accounting for node-level uncertainty is critical for obtaining well-calibrated edge-level confidence estimates. Our experiments on three citation networks demonstrated that our proposed E-ΔUQ methods can substantially reduce the expected calibration error compared to vanilla models. Overall, this work provides novel insights into the importance of node-level uncertainty modeling for link prediction calibration and our proposed stochastic centering framework offers a flexible way to incorporate epistemic uncertainty into existing GNN architectures in a principled manner. An interesting direction for future work is exploring additional auxiliary pretraining objectives to further improve the meaningfulness of node uncertainties.

Table 6.2: **Link Prediction Calibration.**

Dataset	**Method**	**AUPR (↑)**	**ECE (↓)**
Citeseer	E-ΔUQ (v3)	0.8409 ±0.0115	**0.2591** ±0.0178
	E-ΔUQ (v2)	**0.8548** ±0.0076	0.2833 ±0.0075
	E-ΔUQ (v1)	0.8070 ±0.0218	0.3056 ±0.0109
	Vanilla	0.8236 ±0.0115	0.3002 ±0.0062
Cora	E-ΔUQ (v3)	0.8886 ±0.0042	**0.1554** ±0.0060
	E-ΔUQ (v2)	0.8888 ±0.0062	0.1731 ±0.0181
	E-ΔUQ (v1)	0.8598 ±0.0207	0.2640 ±0.0125
	Vanilla	**0.8936** ±0.0066	0.3503 ±0.0146
Pubmed	E-ΔUQ (v3)	0.8775 ±0.0098	0.1818 ±0.0048
	E-ΔUQ (v2)	0.8701 ±0.0016	**0.1538** ±0.0059
	E-ΔUQ (v1)	**0.9069** ±0.0063	0.1801 ±0.0117
	Vanilla	0.8897 ±0.0091	0.1980 ±0.0035

Part III: Large Language Models and Graph Representation Learning

CHAPTER 7

Large Language Model Guided Graph Clustering

7.1 Introduction

While the preceding parts of this thesis focused on improving training protocols for accessing better representation qualities, in Part III, we focus on how combining natural language and structural data can help surpass limitations of the data when performing various graph machine learning tasks. We begin, in this chapter, by studying graph clustering, an unsupervised task which seeks to assign nodes to different clusters such that the resulting assignments capture salient topology and uncover useful concepts, under this setting. Notably, many real-world problems can naturally be formulated as graph clustering, including recommending groups of items in an e-commerce shopping graph or identifying groups of friends in social networks [202, 203, 204]. Most modern, performative clustering methods utilize GNN encoders due to their expressivity [27], scalability, and ability to effectively handle vector-valued node attributes [24, 25].

Recently, however, there has been growing interest in *text-attributed graphs* (TAGs) [75, 76], where natural language text is available as an additional node attribute. Unfortunately, GNNs are not able to directly handle this information rich text and instead utilize semantic embeddings, potentially limiting overall performance. To this end, a variety of (pre/co/joint) training-based [66, 67, 68, 69] and graph specific prompting-based strategies [205, 71, 72, 73, 74] have been recently proposed for using large language models (LLMs) [65, 64] in conjunction with GNNs on *supervised* tasks, e.g., link prediction, node classification, and graph classification, to directly handle this text and take advantage of the LLM's impressive world-knowledge.

While clustering on TAGs could also benefit from joint LLM+GNN methods, it not only remains unclear how to adapt existing supervised approaches for unsupervised graph clustering, but also is prohibitively expensive in many real-world applications due to significant hardware

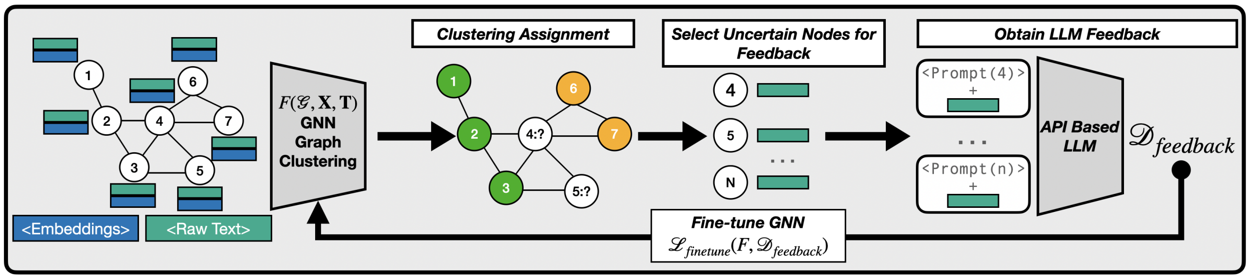


Figure 7.1: **Overview of GCLR.** Given an initial GNN-based graph clustering solution, **F**, GCLR identifies uncertain nodes, obtains LLM guidance through prompting and then fine-tunes the GNN accordingly. Thus, incorporating both graph, world, and semantic (through sentence transformer node attributes) knowledge in clustering assignments.

requirements, incurred through training or hosting LLMs, or API expenditure, incurred by prompting over large sets of nodes. Given that GNN clustering methods are scalable to large graphs by design and have much lighter hardware requirements [206, 78, 77, 207, 208, 209], it is more cost effective to *selectively* use the LLM to *improve* the GNN's initial clustering assignment; thereby limiting the overall expenditure. While a natural framework for such a resource constrained setting is active learning (AL) [210, 63, 211, 212], which selectively queries an expensive oracle for labels to maximize performance under a fixed budget, there are several differences arising from an LLM oracle and the unsupervised nature of graph clustering that must be addressed. Namely, that (i) it is unclear how to select, query, and incorporate LLM feedback to improve GNN clustering solutions, and (ii) the LLM is an *imperfect* oracle, complicating how the model should be updated.

Our Contributions. To this end, we propose GCLR (**G**raph **C**lustering with **LL**M **R**efinement), a flexible active learning framework specifically designed for clustering on TAGS. It uses carefully designed prompting strategies to elicit more reliable and useful feedback for clustering from the LLM and uses simple strategies when fine-tuning to improve tolerance to noisy labels, overall outperforming GNN-only clustering methods. Our contributions are summarized as follows:

• **Eliciting Graph Clustering Feedback from LLMs (Sec. 7.4.1.)** We rigorously study how to obtain feedback from LLMs that is both amenable to clustering and a useful signal for fine-tuning.

• **Incorporating Noisy Feedback from LLMs (Sec. 7.4.2.)** Given the feedback provided by the LLM, we propose training protocols that support fine-tuning deep graph clustering algorithms with imperfect feedback.

• **Extensive Experiments Refining Clustering with GCLR (Sec.7.5)** Across three text-attributed graphs with four different graph clustering algorithms, we demonstrate that

GCLR can improve the graph clustering performance.

7.2 Background & Related Work

In this section, we briefly introduce deep attributed graph clustering and relevant works for combining LLMs and GNNs when working with TAGs. Please see [30] and [213], respectively, for comprehensive surveys.

Deep Attributed Graph Clustering. While unattributed graph clustering has a rich history in network analysis through modularity maximization, spectral clustering, and cuts-based approaches, the success of GNNs in graph representation learning has lead to growing interest in deep clustering methods that efficiently leverage both node-level attributes and topology. Broadly, such methods either (i) learn node representations using a self-supervised or unsupervised objective, and then perform clustering given these representations or (ii) learn both the embeddings and clustering assignments end-to-end through specialized clustering-based losses. While reconstructive [214, 215] and adversarial frameworks [216] were initially popular, in this work, we focus on contrastive [77, 217, 52, 78] and pooling-based methods [209, 207, 208]. Such methods, which, respectively, use contrastive losses to learn discriminative node representations or propose novel pooling layers that optimize for clustering-based losses (e.g., spectral relaxations of modularity or mincut), are more performative, efficient, and scalable than adversarial or reconstructive approaches. Moreover, as we will discuss in Sec. 7.4.2, these methods are more amenable to fine-tuning. Indeed, fine-tuning contrastively pre-trained representations is well-known to induce state-of-the-art performance on a variety of supervised tasks in both vision and graph representation learning.

LLMs + Graphs. Recent approaches that seek to combine graphs/GNNs and natural-language/LLMs can be categorized as being "predictors" (the LLM provides predictions), "encoders" (sentence transformers or other LLMs are used to provide input node features), or "aligners" (GNNs and LLMs jointly trained to perform the task) [213]. Various mechanisms, including prompting [218], fine-tuning [219], variational expectation maximization [67], joint optimization [9], and distillation [220], have been proposed to fulfill these roles, typically on supervised tasks. Instead, GCLR uses the LLM as a refiner and enhancer, as the LLM is only prompted to provide feedback for updating the underlying GNN-based graph clustering solution and sentence transformers are used to provide input node embeddings. This allows us to avoid the expensive fine-tuning of either LLMs or pre-trained language models, as well as exploit the scalability of graph clustering algorithms.

7.3 Problem Formulation

In this section, we formally introduce our problem setting, as well as assumptions and constraints.

Notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{X}, [\mathcal{Y}])$ represent a graph with its respective node set, edge set, raw node-based text information, embedded node attribute information (e.g., some embedding of a node's text), and optional ground-truth cluster assignment. Further, let N be the number of the nodes, M be the number of edges, K be the desired (or ground-truth) number of clusters, d the dimension of the hidden representation, $\mathbf{A} \in \mathbb{R}^{N \times N}$ be the corresponding adjacency matrix, and $\mathbf{X} \in \mathbb{R}^{N \times d}$ be a matrix representation of $\mathcal{X}$.

Problem statement. Let $\mathbf{F} : (\mathbf{A}, \mathbf{X}) \to \mathbf{Z}^{N \times d}$ be a GNN-based encoder that outputs d-dimensional node representations, and $\mathbf{C} : (\mathbf{Z}, K) \to [0, K]^N$ be an embedding-based clustering algorithm, e.g., k-means, pooling layer, where $\mathbf{C}$ may optionally be parameterized and optimized end-to-end with the encoder. Then, the clustering assignments, $\mathbf{K}^{N \times K}$ can be obtained as: $\mathbf{K} = \mathbf{C}(\mathbf{F}(\mathbf{A}, \mathbf{X}), K)$. $\mathbf{K}$ is assumed to be an imperfect assignment, i.e., there exist samples that are mis-assigned to clusters and/or cluster topics are noisy. We seek to use the LLM's world-knowledge and natural language understanding to improve $\mathbf{K}$. Given that $\mathbf{K}$ is already topology-aware due to the GNN encoder and semantic-aware since pre-trained sentence transformers are used to encode the raw text, the LLM provides an complementary source of information. Indeed, the performance of LLMs in zero-shot node classification suggests that their world knowledge is well-suited for graph tasks. We assume pre/co/joint-training is prohibitively expensive and only prompting is available to obtain LLM feedback, and further make the reasonable assumption that there is a limited budget, $\mathbf{B}$, for API calls/prompting. Thus, our objective is to induce the best refined assignment, $\mathbf{K}_{refine}$, while remaining under budget. This problem setting is amenable to active learning, which we introduce conceptually here but note that subsequent sections will discuss how GCLR instantiates AL for clustering.

Active Learning. While much of deep learning is data-intensive and requires large labeled datasets for strong performance, deep active learning seeks to maximize performance in a setting where labels or feedback is expensive to obtain. AL consists of three key components: a *query function*, $\mathbf{Q}$, which determines which samples from the unlabeled data pool should be selected for obtaining feedback, *an oracle*, which provides feedback to create a labeled dataset, $\mathcal{D}_{\text{feedback}}$, and a *training protocol*, which defines a loss, $\mathcal{L}_{\text{feedback}}$, and update procedure for how the model will incorporate said feedback.

Query functions [221, 222, 223] are broadly designed to identify the samples where labeling will have the most impact. Effective functions often use sample uncertainty, difficulty or coverage to select points. The oracle serves as a proxy for an expensive but reliable labeling procedure, for example human annotators or wet-lab experiments. The training protocol is designed to ensure stability, and avoid over-fitting when operating over small batches of data. While some graph AL strategies have been recently proposed, these methods focus on semi-supervised node classification and are not directly applicable to our problem setting.

Moreover, we emphasize that while AL traditionally assumes that (i) the oracle is trustworthy, we do not know apriori the reliability of the LLM's feedback and (ii) our problem is unsupervised, so existing AL query functions, and training protocols may not be well-suited [197, 224, 225]. Lastly, we note that while it is possible to receive dataset-level or task-level feedback, we focus on node-level feedback as it is more scalable for larger graphs (only a subset of nodes will receive feedback), and is more amenable with contrastive and pooling-based graph clustering algorithms, as they already provide node-level embeddings and assignments. In subsequent sections, the design of $\mathbf{Q}$ and $\mathcal{L}_{\text{feedback}}$ for clustering on TAGs is discussed in detail.

7.4 GCLR: Graph Clustering with LLM Refinement

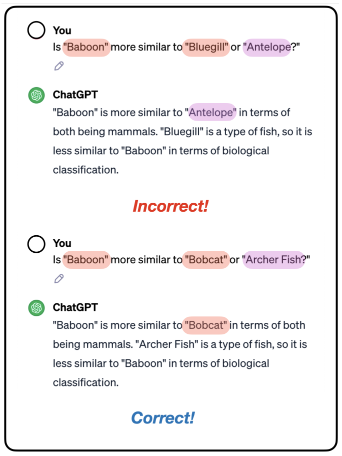
In this section, we formally introduce GCLR, our framework for graph clustering with LLM refinement (Fig. 7.1). We begin by discussing how to obtain useful feedback for graph clustering from LLMs and then present how to identify and refine the initial solution accordingly.

7.4.1 Eliciting Feedback from LLM for Graph Clustering

While feedback in traditional AL typically corresponds to an oracle selecting a label from a predefined set of classes, it is less clear what form the feedback should take when performing clustering. Intuitively, feedback should help improve the similarity of the queried node with the cluster that it belongs to. However, the precise form of the feedback may vary, and its unclear how to prompt the LLM to accurately ascertain this information.

To this end, we discuss the advantages and disadvantages of three different strategies for prompting the LLM to obtain clustering feedback. We begin by discussing a recently proposed strategy for LLM guided text clustering.

Triplet-Based Prompting. ClusterLLM [226] is a recently proposed state-of-the-art LLM guided *text* clustering method that first selects uncertain samples (e.g., queries), $\mathcal{Q}_i$, and two random samples from each query's two nearest clusters, and then prompts the LLM to predict which of the two samples is "more similar" to $\mathcal{Q}_i$; the more similar sample is considered


You
Is "Baboon" more similar to "Bluegill" or "Antelope?"

ChatGPT
"Baboon" is more similar to "Antelope" in terms of both being mammals. "Bluegill" is a type of fish, so it is less similar to "Baboon" in terms of biological classification.

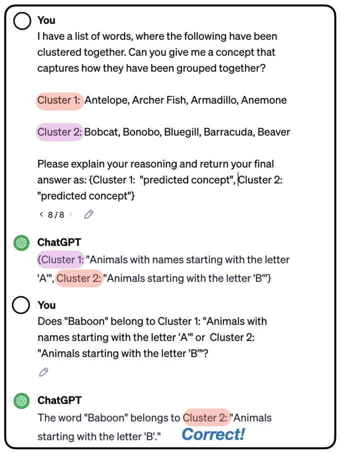
Incorrect!

You
Is "Baboon" more similar to "Bobcat" or "Archer Fish?"

ChatGPT
"Baboon" is more similar to "Bobcat" in terms of both being mammals. "Archer Fish" is a type of fish, so it is less similar to "Baboon" in terms of biological classification.

Correct!

(a) **Triplet Based Feedback**


You
I have a list of words, where the following have been clustered together. Can you give me a concept that captures how they have been grouped together?

Cluster 1: Antelope, Archer Fish, Armadillo, Anemone

Cluster 2: Bobcat, Bonobo, Bluegill, Barracuda, Beaver

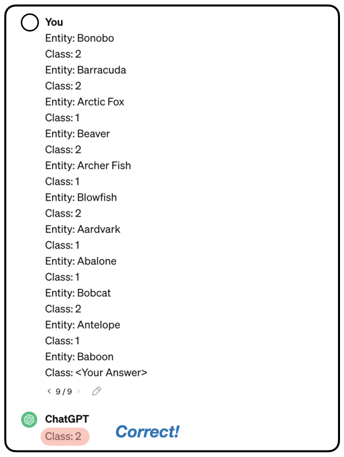
Please explain your reasoning and return your final answer as: {Cluster 1: "predicted concept", Cluster 2: "predicted concept"}

ChatGPT
{Cluster 1: "Animals with names starting with the letter 'A'", Cluster 2: "Animals starting with the letter 'B'"}

You
Does "Baboon" belong to Cluster 1: "Animals with names starting with the letter 'A'" or Cluster 2: "Animals starting with the letter 'B'"?

ChatGPT
The word "Baboon" belongs to Cluster 2: "Animals starting with the letter 'B'." *Correct!*

(b) **Concept Based Prompt**


You
Entity: Bonobo
Class: 2
Entity: Barracuda
Class: 2
Entity: Arctic Fox
Class: 1
Entity: Beaver
Class: 2
Entity: Archer Fish
Class: 1
Entity: Blowfish
Class: 2
Entity: Aardvark
Class: 1
Entity: Abalone
Class: 1
Entity: Bobcat
Class: 2
Entity: Antelope
Class: 1
Entity: Baboon
Class: <Your Answer>

ChatGPT
Class: 2 *Correct!*

(c) **In-Context Based Prompt**

Figure 7.2: **Example of LLM Feedback.** Using the example graph in Fig. 7.3, we prompt `chat-gpt-3.5-turbo` with different strategies to demonstrate the importance of aligning the LLM's and GNN's implicit similarity functions to obtain valid feedback. Indeed, we see that triplet-based prompting can be unreliable as it does not allow the LLM to infer the underlying similarity. For example, with the query, "Baboon" with triplets containing the land animals from from Cluster 1 (starts with B) and aquatic animals from Cluster 2, the LLM assigns Baboon to cluster 1 (Baboon, Bobcat, Archer Fish), which is consistent with the graph solution. However, when we prompt `chat-gpt-3.5` with a triplet containing *aquatic* animals from Cluster 1 and *land* animals from Cluster 2 (Baboon, Bluegill, Antelope), the LLM assigns the query to Cluster 2 as it is also a land animal. In contrast, we find that both concept-based and incontext-based prompting are able to correctly infer the GNN's similarity function and provide valid feedback.

a "positive" sample and the other is a "negative" sample. Here, $\mathcal{D}_{\text{feedback}}$ corresponds to the set of triplets (query, positive, negative) determined by the LLM and $\mathcal{L}_{\text{feedback}}$ is InfoNCE. While such an approach can conceptually be applied to graph clustering, there are some limitations.

Insofar as clustering requires learning a similarity function that can be used to partition samples into meaningful groups, it is important that the oracle is aware of this function so the resulting feedback is aligned to existing the partitioning. In text clustering, since both the encoder (BERT, E5, etc) and the larger, oracle LLM (Chat-GPT, Llama) are text based models, they share a similar prior for this similarity function. In contrast, when performing graph clustering, the GNN incorporates topological information unavailable to the LLM and may utilize a different function that the LLM. Indeed, in Fig. 7.3, we construct a simple synthetic example where the GNN and LLM utilize different similarity functions to identify concepts by design. We observe, in Fig. 7.2a, that the oracle (`chat-gpt-3.5-turbo`) provides *un*reliable feedback when the triplet prompt contains random samples that do *not* overlap with the GNN's similarity function, but is reliable when the random samples are selected to align with the LLM's implicit similarity function.

Finally, we note that the performance of triplet-based feedback is closely tied to the quality of the initial clustering solution, artificially handicapping the LLM's performance. Given that the initial clustering solution is imperfect, randomly selecting samples from the two closest clusters can create triplets that do not actually represent the corresponding clusters, leading the LLM to perform a meaningless selection. Moreover, there is a loose upper-bound of the triplet formulation as the queries' "correct" cluster must be within the top-2 closest clusters. If this is not the case, the LLM will necessarily have to respond to an ill-formed triplet and will provide incorrect feedback. Due to the rapidly increasing capabilities of LLMs, it is possible that future LLMs will achieve perfect performance on valid triplets, however, the error incurred by ill-formed triplets is irreducible.

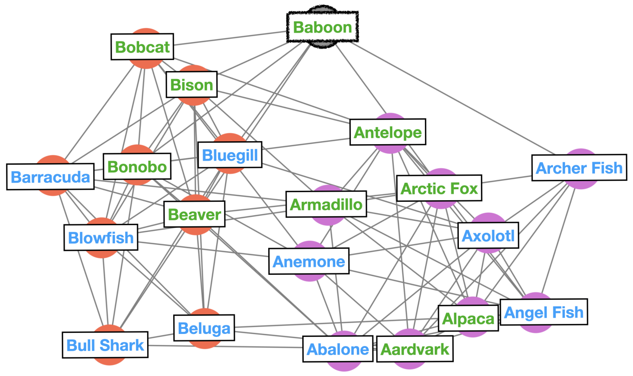


Figure 7.3: **Unaligned Notions of Similarity.** The following stochastic block model graph has clusters that correspond to whether a particular animal's name begins with **"A"** or **"B."** However, an alternative clustering according to **"land"** vs. **"aquatic"** animals is also valid and more semantically interesting. Indeed, when GPT-3.5 is asked whether a "Baboon" is more similar to a "Bluegill" or "Antelope," it replies with "Antelope" as it is also a land mammal. This emphasizes that (i) simple pairwise comparisons may not be sufficient for providing feedback and (ii) LLMs and GNN clustering algorithms may utilize disparate notions of similarity.

In-Context Similarity Learning. As discussed above, it is critical that the LLM can infer the similarity function implemented by the GNN. Given the impressive in-context learning capabilities of LLMs [227, 228], we consider a prompt that allows the LLM to directly infer it by providing several examples of the node's raw text and their corresponding cluster IDs, and the text of the unlabeled query (See Fig. 7.2b for an example.) Here, the LLM can be seen as performing a prediction task amongst pseudo-labels defined by the initial clustering, where $\mathcal{D}_{\text{feedback}} = \{([0,\ldots K]|i \in \mathcal{Q}\}$. We note that the choice of $\mathcal{L}_{\text{feedback}}$ is flexible and discuss it in detail later. Notably, by ensuring that the prompt contains samples from all clusters, the LLM can (i) more holistically infer what concepts underlie clusters and (ii) predict an assignment for a query that does not belong to the top-2 clusters. This allows us to circumvent the previous issue where the upper-bound on refined performance was restricted by the number of samples where the preferred assignment was contained in the top-2 clusters.

However, directly inferring the similarity function from in-context examples becomes more difficult as the number of clusters grows as (i) the number of exemplars must correspondingly reduce to remain within the context length and (ii) if the number of clusters is sufficiently large, it is not possible to provide exemplars from all clusters. Furthermore, the selection and ordering of exemplars can have a significant impact of the LLM's ability to correctly predict a query's assignment, leading to potential loss of performance during fine-tuning.

Concept-based Prompting. To avoid the aforementioned issues with incontext-prompting, we draw inspiration from topic modeling [229, 230] and design an additional ”concept-based” prompting strategy where we first prompt the LLM to infer the concepts that were used to group samples and then create a prediction task where the LLM is prompted to select amongst the generated concepts. (See Fig. 7.2b for an example.) To generate concepts, we provide the LLM samples from each cluster and ask it to provide a ”title” and ”short description” that explains how these samples are grouped together. These generated titles and descriptions are then provided as options for the LLM to identify the most similar cluster for a particular query. Notably, by providing the titles/descriptions of all clusters, we can avoid the upper-bound encountered by triplets while simultaneously allowing the LLM to at least partially infer the GNN's similarity function.

Experimental Setup. We verify the effectiveness of the proposed feedback elicitation strategies on several public graph datasets, where the provided node labels serve as ground-truth cluster labels. `mixtral-8x-7b` is used as the oracle, and four different graph clustering backbones are used to obtain the initial clustering solutions. We sort the samples according to the entropy of the distance to the two nearest clusters (a proxy for sample difficulty) and prompt the LLM for each sample as per the discussed strategies. The accuracy of the LLM's solutions with respect to the ground-truth solution is reported, where the Hungarian algorithm to align clusters to labels. Please see App. D.2 for example prompts, comprehensive experimental details and dataset statistics.

Results. The following observations are made from Table 7.1. We observe that across datasets and clustering methods, that the “concepts” strategy is the best or second best performing prompting strategy most often. While In-Context prompting achieves comparable performance on some datasets, we note that it is significantly more expensive. Indeed, every InContext prompt contains multiple exemplars per cluster, while “concepts” only processes these exemplars once to obtain the generated titles and descriptions, which are then directly used in the prompt. “Triplets” is the cheapest strategy in terms of token length, but lags behind on performance, failing to achieve the best performance on any dataset. Lastly, we note that the GNN outperforms the LLM on full dataset (100th percentile) accuracy on 9/12

Table 7.1: **Reliability of LLM as an Annotator.** The accuracy of the GNN-based clustering solution and three prompting strategies are reported at the 10\50\100-th most difficult percentile of the dataset. The best performance overall is **bolded**, while any prompting-based method is colored if it exceeds the accuracy of the GNN, and the 2nd best prompting based method is underlined.

Dataset	**Method**	**GNN**	**Concepts**	**Incontext**	**Triplets**
		Graph Only	LLM Only		
citeseer	diffpool	32.1\36.2\49.7	**36.2** \41.1 \49.1	34.6 \36.2 \46.7	29.2\34.1\44.0
	dinknet	40.6**54.7****70.3**	30.8\32.9\47	**48.7**\48.3\59.6	43.1\50.6\62.1
	dmon	36.5\38.2\44.1	**40.9****39.9**\43.9	36.2\37.7\42.9	36.8\38\41.7
	mincut	35.8**52.2****66.5**	38.4\46.1\58.5	**42.1**\50.5\60.5	34.3\46.5\57.1
cora	diffpool	32.6**40****54.7**	**35.6**\36.0\37.7	34.4\36.6\50.2	33.7\36.9\48.8
	dinknet	**37.4****50.7****65.8**	32.2\36.8\39	24.8\36.0\52.7	35.2\47\58.2
	dmon	42.6**52.4****60.9**	36.3\41.4\40.7	**46.3**\51.3\56.9	40\47.9\54
	mincut	40**53.6****68.4**	42.2\46.5\55.7	**43.7**\50.5\63.3	37.8\49.8\60.9
wikics	diffpool	25.5\32.2\48.3	**36.0****40.4****52.7**	33.9\37.1\47.9	25.9\30.8\44.2
	dinknet	37.7\51.2\66.5	**51.2****56.5**\64.8	35.8\36.9\51.1	35.0\44.5\54.8
	dmon	28.1\31.2\36.9	**55.2****55.2****57.2**	39.9\41.3\41.3	28.7\31.2\35.8
	mincut	36.5\24.4\26.9	31.9**29.6**\29.8	**37.5**\27.9**31.0**	32.4\24.1\25.2

settings, indicating that, in addition to being prohibitively expensive, prompting the LLM for every node would not be as effective as the initial GNN solution. Indeed, there are several situations where the LLM's feedback is less effective than the GNN's, highlighting that care must be taken when updating the GNN.

7.4.2 Refining GNN-Based Clustering with Feedback

While the proposed prompting strategies help improve the LLM's feedback, we must now incorporate this imperfect feedback into the GNN to scalably improve the overall clustering solution. Indeed, even with oracle feedback, it is not immediately guaranteed that the training dynamics of different graph clustering approaches will be readily refined and induce to better solutions. We begin by discussing at a high-level how to fine-tune different types of graph clustering frameworks.

Finetuning Setup. While reconstructive [214, 215] and adversarial frameworks [216] were initially popular for graph clustering, we focus on more recent contrastive [77, 217, 52, 78] and pooling-based methods [209, 207, 208] as they are more scalable and performative. Furthermore, there is extensive literature on fine-tuning contrastively pre-trained models (typically for supervised tasks) that we can leverage when defining $\mathcal{L}_{feedback}$. Indeed, both in-context and concept-based prompting induce a dataset, $\mathcal{D}_{feedback} = \{([0, \ldots K]|i \in \mathcal{Q}\}$, that consists of queried nodes and their predicted cluster assignments. Thus, we can consider refinement as a supervised task with LLM-provided pseudo-labels.

When working with pooling-based methods (DMon, MinCut, and DiffPool, etc), $\mathbf{F}$ directly predicts the cluster assignment as the node features are pooled to the number of clusters. For contrastive methods like DinkNet, we can initialize a classifier using parameterized cluster centers or those obtained using KMeans. Then, given the classifier and $\mathcal{D}_{feedback}$, we can naturally define $\mathcal{L}_{feedback}(\mathcal{D}_{feedback}, \mathbf{F})$ using the cross-entropy loss. While other losses (triplet, InfoNCE, SupCon) are certainly possible, we empirically find that cross-entropy is effective. However, since $\mathcal{D}_{feedback}$ is expected to contain incorrect labels, but the error-generating process is unknown, naively training on the labels may diminish performance. Thus, we consider the following simple but effective strategies for improving the finetuning performance.

Strategies for Handling Noisy Labels. Given that our prompting strategies induce a classification task, we consider using the model's predicted confidence in order to eliminate potentially noisy labels. Namely, we compute the LLM's confidence in its predictions by obtaining log-probability of the top-2 tokens corresponding to cluster predictions. Alternative prompting strategies and specialized losses have been proposed for better calibration [231, 232, 233] but we do not consider them due to their additional expense.

To further stabilize and improve training, we consider augmenting $\mathcal{D}_{feedback}$ with samples well-clustered by the GNN, where probits of the predicted clusters are used to identify confident assignments. The loss is computed separately for the LLM-labeled and GNN-labeled samples, and aggregated as $\alpha\mathcal{L}_{finetune,LLM} + \beta\mathcal{L}_{finetune,GNN}$, where α and β are constrained to be a convex combination. By varying α and β, we can express different levels of certainty in the feedback. Since the optimal weighting is not known apriori, creating a simple deep ensemble [179] by varying α, β to train multiple independent models may further improve performance. Though this incurs additional training expenditure, it is not substantial with respect to training the initial model as clustering losses often approximate quadratic operations, or obtaining feedback. We assess the effectiveness of each of these components and GCLR as a whole in the following section. Additionally, we use a simple ensembling strategy to further stabilize performance.

Namely, the loss is computed separately for the LLM-labeled and GNN-labeled samples, and then computed as $\alpha\mathcal{L}_{finetune,LLM} + \beta\mathcal{L}_{finetune,GNN}$, where α and β are constrained to be a convex combination. By varying α and β, we can express different levels of certainty when updating the model. However, as we do not know a priori the optimal weighting, we create a DeepEns that samples D different α, β to train D independent models. To reduce computational and memory costs, only the classifier or a limited portion of the GNN encoder is updated and stored. We emphasize that our training expenditure is not substantial with respect obtaining the initial model as clustering losses often approximate quadratic

Table 7.2: **LLM Labels Provide Complementary Information For Active Learning.** To understand how GCLR improves clustering, we compare the performance of different feedback mechanisms (None, GNN pseudo labels, LLM feedback) and finetuning losses (triplet vs. cross entropy). We observe that (i) while both LLM (9/12 Acc.) and GNN (10/12 Acc) feedback generally improves performance over the initial starting solution, that LLM feedback with the cross entropy loss achieves the best accuracy overall (8/12), though performance on intrinsic metrics is more mixed; (ii) on Cora, where GNN feedback was more reliable than LLM feedback, we see understandably, that using the GNN pseudo labels is more effective; (iii) in contrast, on WikiCS, where LLM feedback is much more reliable, we see dominant performance by LLM feedback with cross entropy loss; and (iv) we see see that the cross entropy loss (9/12 Acc., 7/12 Modularity, 7/12 NMI) is more effective than the triplet for finetuning with LLM feedback. Overall, our results demonstrate there is value to refining with LLM feedback. We note that GCLR's performance can further be improved with confidence filtering (Table 7.4) and ensembling (Table 7.3).

		(starting performance) \ GNN Feedback + Cross Ent. Loss \ LLM Feedback + Triplet Loss \ LLM Feedback + Cross Ent. Loss					
Dataset	**Method**	**Acc.** (↑)	**NMI** (↑)	**F1** (↑)	**ARI** (↑)	**COND** (↓)	**MOD** (↑)
citeseer	diffpool	(47.09) \54.69 \48.05 \58.96	(25.59) \25.94 \21.50 \26.84	(23.08) \23.57 \14.65 \19.70	(43.09) \43.33 \33.22 \41.41	(0.23) \0.23 \0.25 \0.24	(0.56) \0.56 \0.45 \0.50
	dinknet	(66.46) \66.43 \67.36 \67.40	(43.08) \43.30 \19.16 \36.97	(42.43) \41.30 \16.37 \27.16	(60.39) \60.58 \42.49 \47.91	(0.07) \0.07 \0.29 \0.09	(0.70) \0.70 \0.51 \0.62
	dmon	(47.89) \49.85 \48.75 \49.87	(28.49) \28.77 \27.11 \27.12	(24.29) \24.61 \18.86 \14.46	(43.65) \43.71 \34.14 \29.87	(0.19) \0.19 \0.25 \0.15	(0.60) \0.60 \0.45 \0.47
	mincut	(64.18) \66.70 \69.82 \67.51	(44.41) \46.21 \40.48 \39.60	(41.95) \43.25 \38.54 \35.81	(61.72) \62.11 \59.54 \59.81	(0.08) \0.09 \0.13 \0.17	(0.73) \0.73 \0.67 \0.64
cora	diffpool	(59.97) \63.6 \43.38 \51.35	(43.46) \42.70 \20.97 \22.21	(36.58) \35.65 \7.83 \6.49	(56.76) \55.64 \29.3 \29.05	(0.24) \0.25 \0.38 \0.32	(0.60) \0.60 \0.33 \0.34
	dinknet	(68.26) \66.84 \67.32 \65.16	(51.98) \50.87 \25.01 \23.42	(44.21) \40.50 \15.16 \9.25	(62.09) \59.20 \41.86 \27.40	(0.12) \0.11 \0.30 \0.08	(0.70) \0.67 \0.49 \0.29
	dmon	(57.56) \60.27 \59.06 \56.70	(41.60) \42.24 \30.18 \30.06	(33.76) \34.64 \20.66 \13.67	(50.94) \51.40 \39.44 \29.40	(0.27) \0.26 \0.38 \0.12	(0.56) \0.58 \0.42 \0.33
	mincut	(64.17) \66.63 \59.91 \61.62	(48.92) \48.92 \39.74 \41.61	(40.35) \40.35 \29.43 \30.54	(58.33) \58.33 \47.28 \54.01	(0.14) \0.14 \0.21 \0.28	(0.70) \0.70 \0.56 \0.54
wikics	diffpool	(43.15) \49.69 \55.44 \58.03	(26.27) \26.36 \37.20 \35.03	(18.87) \19.50 \31.10 \26.28	(39.88) \39.70 \41.12 \46.48	(0.34) \0.35 \0.30 \0.34	(0.48) \0.47 \0.36 \0.44
	dinknet	(66.80) \73.65 \67.48 \74.00	(49.00) \51.84 \47.49 \51.25	(47.80) \53.04 \46.18 \51.57	(56.23) \63.06 \56.97 \63.06	(0.23) \0.21 \0.28 \0.23	(0.55) \0.55 \0.52 \0.54
	dmon	(38.60) \39.68 \43.28 \51.87	(27.47) \27.49 \29.33 \32.51	(20.55) \20.65 \27.48 \31.04	(34.02) \34.18 \34.48 \36.49	(0.48) \0.47 \0.42 \0.26	(0.33) \0.33 \0.33 \0.31
	mincut	(24.70) \32.84 \38.52 \46.36	(6.14) \8.32 \17.99 \16.52	(-0.37) \-0.32 \18.02 \4.8	(7.91) \8.45 \24.71 \24.36	(0.04) \0.04 \0.45 \0.47	(0.03) \0.05 \0.30 \0.27

operations, for example modularity maximization. Lastly, we note that hyper-parameter tuning or early-stopping on unsupervised problems can be difficult as there is not necessarily a validation set available. To this end, we use the conductance, an extrinsic, label free metric, throughout our experiments to tune the learning rate and perform early stopping. We assess the effectiveness of each of these components and GCLR as a whole in the following section.

7.5 Experiments

In this section, we verify the effectiveness of GCLR in refining graph clustering solutions across several public datasets with different graph clustering algorithms.

Experimental Setup. Our set-up is as follows. *Baselines.* We consider the following graph clustering baselines: MinCutPool [207], DMoN [209], DiffPool [208], and DinkNet [77]. *Metrics.* As we use public datasets with available ground-truth clustering, we report accuracy, Normalized Mutual Information, F1, and Adjusted Rand-Index between the predicted and labeled clusters. We intrinsically assess the clustering quality using conductance and modularity (see App D.3 for their precise definitions). *Datasets.* We provide the dataset statistics in Table D.4. *Training.* Both the initial GNN and subsequently finetuned models are trained with early-stopping and the learning rate is tuned amongst 1e-4 and 1e-3.

GCLR. Unless otherwise noted, we use `mixtral-8x-7b` as the oracle LLM and seek feedback on at most 10% of the nodes in the dataset. The query function, $\mathbf{Q}$, is defined to select nodes according to prediction entropy [222]. Here, high entropy nodes are less well-clustered, and labeling them would provide useful information. α and β are both set to 0.5, unless otherwise noted. Results are averaged over 10 seeds.

Results. We begin by confirming that the LLM provides valuable information through its feedback by demonstrating, in Table. 7.2, that subsequent finetuning not only improves performance over the starting clustering solution but also over finetuning on GNN pseudo labels, when reliable. Additionally, we find that using the cross entropy loss is more effective than the triplet loss when finetuning using the LLM feedback. This is in contrast to ClusterLLM, which focused on triplets. Overall, this suggests that GCLR does provide a viable strategy for improving performance clustering performance.

Observation 2. Next, we seek to understand how filtering samples according to confidence can improve GCLR's performance.

We do note that both the GNN and LLM feedback are not guaranteed to be calibrated, but nonetheless empirically find their confidences useful. In particular, in Table. 7.4, we set $\alpha = 0.5$ and $\beta = 0.5$, and consider 2 different filterings : one where the GNN's confidence interval is high and the other where the LLM's confidence interval is high. We find that updating the model usin only high confidence LLM feedback (80th percentile) and GNN feedback at lower percentile improves the accuracy 8/12 times. We posit that the relatively large set of low confidence GNN samples help stabilize training, while the high confidence LLM feedback helps enhance the overall clustering solution.

Observation 3. In settings where the LLM's feedback is less reliable than the GNN's, it is possible to harm the initial clustering solution when updating the intial clustering solution. For example, in Table 7.1, on Cora, the LLM's feedback is less reliable than the GNN's, and in Table 7.2, we see finetuning on GNN feedback leads to better performance than the LLM's. However, we note that even if the LLM's feedback is unreliable it may still contain valuable information. To this end, we create a simple deep ensemble that captures different levels of certainity in either source's feedback by varying α and β when aggregating the loss. In particular, we train 5 different models, where we sample $\alpha \in [0, 0.1, \ldots 0.5]$ and $\beta \in [0.5, 0.6 \ldots 1]$ at evenly spaced intervals. In Table 7.3, we show that using this ensemble can improve performance over a single model where $\alpha = \beta = 0.5$, and see that GCLR improves over the initial clustering solution as desired.

Observation 4. While the above experiments identify query samples according to their entropy, other query functions are viable. In Table. D.1, we consider the following alternative

Table 7.3: **Ensembling Improves Performance with Unreliable Feedback.** Even when the LLM feedback's is unreliable relative to the GNN's, it can still be valuable as there may be samples where the LLM corrects the GNN's misclustered samples. However, as we do not know beforehand how reliable either feedback source is, we create a deep ensemble by sampling different α and β to simulate different levels of confidence in each ensemble source. On Cora, where the LLM's feedback is known to be unreliable, we find that ensembling improves the performance of over a single model where $\alpha = 0.5$ and $\beta = 0.5$, and surpasses the performance of the starting solution as desired. Overall, this indicates that GCLR can help improve the initial clustering solution even with unreliable feedback.

Method	**Ens?**	**Acc.**	**NMI**	**F1**	**ARI**	**COND**	**MOD**
	starting	59.97	43.36	36.58	56.76	0.24	0.60
diffpool	✗	51.35	22.21	6.49	29.05	0.32	0.34
	✓	**61.88**	**45.74**	**38.97**	**58.20**	**0.22**	**0.62**
	starting	68.26	51.98	44.21	62.09	0.12	**0.70**
dinknet	✗	65.16	23.42	9.25	27.40	**0.08**	0.29
	✓	**69.36**	**52.66**	**45.28**	**63.12**	0.12	**0.70**
	starting	57.56	41.60	33.76	50.94	0.27	0.56
dmon	✗	56.70	30.06	13.67	29.40	**0.12**	0.33
	✓	**60.60**	**43.25**	**37.60**	**52.41**	0.24	**0.58**
	starting	64.17	48.92	40.35	58.33	**0.14**	**0.70**
mincut	✗	61.62	41.61	30.54	54.01	0.28	0.54
	✓	**64.63**	**48.96**	**40.77**	**58.79**	**0.14**	**0.70**

query functions: random sampling, sampling the least confidence queries, and sampling queries with the smallest margin between the top-2 predicted clusters. While random sampling incurs some loss in performance, we find that margin sampling performs similarly to entropy sampling and sampling according to least confidence improves performance in some cases.

Table 7.4: **Effect of Confidence Filtering.** While we do not know the reliability of either the LLM or GNN's feedback apriori, we can use their confidence to select samples where the feedback is more likely to be reliable to avoid finetuning on misleading samples. Here, we filter samples based on the ascending confidence percentile, so the 80th percentile corresponds to samples whose confidence is greater than or equal to 80% of total samples. We observe that filtering improves performance without filtering (11/24 Acc.) and over the starting (no finetuning) solution (17/24 Acc.). In particular, 80% LLM and 20% GNN filtering improves performance over no filtering (8/12 NMI, 10/12 Mod.) On WikiCS, no filtering performs the best, suggestive of the LLM's better reliability. Best performance is **bolded** and accuracy of the starting solution is in parentheses.

Dataset	**Method**	**LLM**	**GNN**	**Acc.**	**NMI**	**F1**	**ARI**	**COND**	**MOD**
citeseer	diffpool (47.09)	20	80	53.04	22.67	15.06	34.93	0.31	0.45
		80	20	56.71	**26.94**	**23.18**	**41.90**	**0.21**	**0.56**
		0	0	**58.96**	26.84	19.70	41.41	0.24	0.50
	dinknet (66.35)	20	80	67.61	38.14	32.03	50.99	**0.08**	0.64
		80	20	**67.43**	**40.23**	**37.88**	**56.47**	0.10	**0.67**
		0	0	67.40	36.97	27.16	47.91	0.09	0.62
	dmon (47.89)	20	80	**51.21**	26.85	18.27	31.64	**0.15**	0.50
		80	20	51.14	**30.06**	**25.30**	**41.72**	0.17	**0.59**
		0	0	49.87	27.12	14.46	29.87	**0.15**	0.47
	mincut (64.17)	20	80	61.42	31.79	26.94	47.84	0.26	0.56
		80	20	65.40	**41.32**	**38.01**	59.37	**0.13**	**0.69**
		0	0	**67.51**	39.60	35.81	**59.81**	0.17	0.64
cora	diffpool (59.97)	20	80	55.28	29.53	16.07	39.33	0.39	0.39
		80	20	**61.94**	**41.64**	**36.77**	**55.67**	**0.27**	**0.57**
		0	0	51.35	22.21	6.49	29.05	0.32	0.34
	dinknet (66.20)	20	80	67.15	36.21	24.09	42.83	0.13	0.50
		80	20	**67.87**	**48.03**	**36.82**	**52.04**	0.12	**0.66**
		0	0	65.16	23.42	9.25	27.40	**0.08**	0.29
	dmon (57.55)	20	80	58.07	36.72	24.99	40.19	0.23	0.47
		80	20	**62.06**	**41.79**	**35.56**	**50.52**	0.25	**0.57**
		0	0	56.70	30.06	13.67	29.40	**0.12**	0.33
	mincut (64.17)	20	80	61.04	38.40	28.22	48.95	0.34	0.50
		80	20	**64.55**	**47.15**	**38.89**	**57.82**	**0.19**	**0.65**
		0	0	61.62	41.61	30.54	54.01	0.28	0.54
wikics	diffpool (43.34)	20	80	51.53	27.52	17.87	**37.83**	0.41	0.39
		80	20	50.60	24.03	16.68	34.87	0.40	0.42
		0	0	**58.03**	**35.03**	26.28	46.48	**0.34**	**0.44**
	dinknet (71.25)	20	80	66.51	45.90	41.76	54.10	0.26	0.53
		80	20	66.79	48.39	41.85	55.66	**0.23**	**0.54**
		0	0	**74.00**	**51.25**	**51.57**	**63.06**	0.23	**0.54**
	dmon (37.515)	20	80	42.81	27.14	19.03	30.33	0.37	0.29
		80	20	40.92	28.11	20.24	32.39	0.46	**0.33**
		0	0	**51.87**	**32.51**	**31.04**	**36.49**	**0.26**	0.31
	mincut (24.70)	20	80	42.79	**19.16**	**7.50**	17.07	**0.27**	0.23
		80	20	43.58	14.74	3.77	19.57	0.30	0.14
		0	0	**46.36**	16.52	4.80	**24.36**	0.47	**0.27**

Observation 5. While the above experiments identify query samples according to their entropy, other query functions are viable [222, 223]. In Table. D.1, we consider the following alternative query functions: random sampling, sampling the least confidence queries, and sampling queries with the smallest margin between the top-2 predicted clusters. While random sampling incurs some loss in performance, we find that margin sampling performs similarly to entropy sampling and sampling according to least confidence actually improves performance in some cases.

Observation 6. Traditional active learning generally benefits from increasing the labeling budget as the oracle provides additional reliable feedback. In contrast, we find in Table. D.2 that increasing the budget does not have a substantial impact on performance. We believe this is partially due to an imperfect oracle and the bootstrapping that occurs from stabilizing training with GNN provided pseudo-labels.

7.6 Conclusion

In this work, we proposed GCLR to improve graph clustering solutions on text attributed graphs by eliciting feedback from LLMs. In order to avoid large prompting expenditure, GCLR actively queries the LLM on only on uncertain nodes and uses various prompting strategies to obtain clustering feedback. This feedback is then used to update the initial GNN based clustering solution. Since LLM and GNN feedback can be unreliable, confidence filtering and ensembling are used to further improve performance. Given that GCLR's efficacy is constrained by the quality of the LLM provided feedback, future directions of work include designing prompting/training strategies to improve the reliability of the LLM oracle.

CHAPTER 8

Exploring the Robustness of LLM+GNN models on text-attributed graphs

8.1 Introduction

Having discussed graph clustering on text-attributed graphs in the preceding chapter, we turn our attention to the equally important task of node classification on TAGs. Indeed, LLMs [64, 65] are increasingly being combined with GNNs [66, 67, 68, 69, 70] to perform to node classification in this setting [75, 76], due to their impressive world knowledge, reasoning and natural language understanding capabilities [234, 235, 236, 237, 238, 239, 240, 241, 242]. While such joint approaches generally report improved performance, e.g., better node classification accuracy, this is a coarse-grained analysis of a model's generalization ability and the mechanisms by which LLM+GNN models makes their predictions remain relatively under explored. Indeed, GNNs, like vision models, are well-known to be susceptible to adversarial attacks [33], i.e. imperceptible input perturbations designed to decrease performance. However, to the best of our knowledge, the robustness of LLM+GNNs has been relatively under studied with respect to (i) traditional structural or node attribute attacks [36, 37] or (ii) natural language perturbations to text attributes [243], a new avenue of attack for joint models. To this end, this chapter rigorously explores the robustness of different types of LLM+GNN to both attack avenues, and outlines several interesting avenues for further research.

8.2 Background & Related Work

In this section, we briefly discuss relevant background on graph learning for text-attributed graphs, graph adversarial attacks and textual attributes. For an in-depth discussion, please see the following surveys respectively: [213], [244] and [245].

8.2.1 LLM+GNN Models

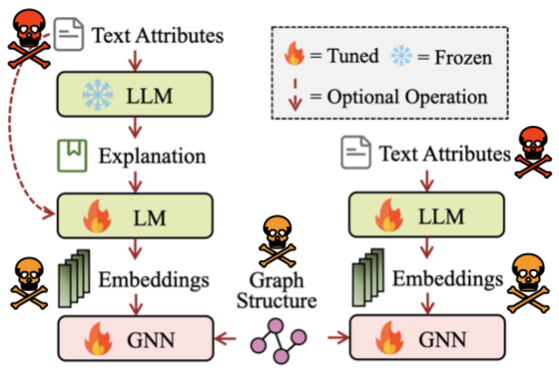
Recent works on LLM+GNN models can be broadly categorized into the following taxonomy [9]: (i) *enhancers*, which use both LLMs and smaller pretrained language models (PLMs) to obtain better node attribute embeddings prior to training the GNN [70, 246, 247], (ii) *predictors*, which provide the LLM with textual and structural information in order to directly make predictions[71, 73, 72] and (iii) *aligners*, which aims to establish a jointly aligned subspace that can be used for other tasks [67, 248, 69]. Insofar as aligners often require paired data, most LLM+GNNs focused on node classifications are either enhancers or predictors.

For example, [205] recently proposed TAPE, an enhancer consists of prompting, embedding and training phases. First, TAPE prompts an LLM (Llama-2, GPT) to obtain node label prediction and corresponding explanation. Then, a PLM (DeBerta [249], RoBerta [250]) is finetuned on a node classification loss to obtain embeddings for the explanation, prediction and original text. Finally, the embeddings extracted from the PLM are used to train a GNN to make predictions. Notably, the obtained node embeddings can substantially improve performance over previously used "shallow" embeddings, such as bag-of-words [4] or TF-ID [251], by infusing semantic awareness into the model.

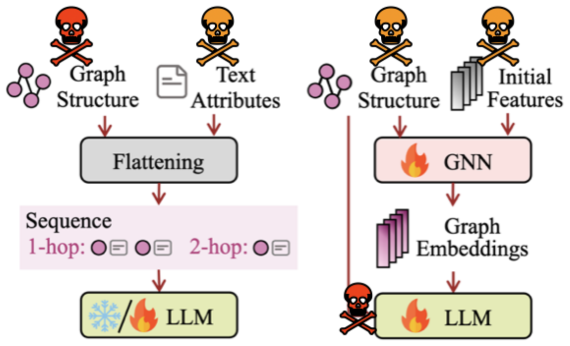
In contrast, predictors must ensure that the input to the LLM contains structural information as the LLM will directly make the final prediction. As an example, [73] recently proposed GPT4GRAPH, which uses the graph markup language and summaries over neighborhood node attributes to capture topological information when prompting the LLM for a prediction. Finding an appropriate natural language graph representation is an open challenge and several other strategies have been proposed including natural narration, alphabetizing node ids, and creating graph syntax trees [72]. On datasets with strong textual attributes, predictors have impressive zero-shot performance relative to traditional GNNs. However, due to limited context length, it can be challenging to include multi-hop information or capture long-range dependencies. This issue will become less problematic as LLMs which support larger windows are developed.

8.2.2 Graph Adversarial Attacks

Like their image counterparts, graph adversarial attacks seek to create "imperceptible" perturbations which negatively effect performance [36, 37]. Attacks are considered targeted when the goal is to misclassify a particular node, and untargeted when the goal is to decrease performance overall. Attacks are further categorized as poisoning or evasion , depending on if the perturbed input is used during or after training, respectively. Typically, structural attacks are constrained to modify only a portion of the overall edges, as a proxy to imperceptibility.



(a) **LLM-As-Enhancer**



(b) LLM-As-Predictor

Figure 8.1: **LLM+GNN Methods Offer New Avenues of Vulnerability.** While combining LLMs and GNNs for tasks on text-attributed graphs has led to the improved performance, we note that both LLM-As-Enhancers and LLM-As-Predictors contain new avenues for adversarial attacks. Here, we show traditional GNN-only or LLM-only avenues in orange, and highlight new avenues in red. (Figured adapted from [9]).

It remains an open question if such a constraint is able to preserve the semantics of the original class label [43]. Indeed, if semantics are altered and the classifier correctly predicts the new semantically consistent label, then the perturbed sample can no longer be considered adversarial. In addition to structural attacks, it is possible to attack the node attributes, where perturbations are restricted to some normed ball. In the case of untargeted attacks, the accuracy over the entire test set should be maximally harmed, while in targeted attacks, the goal is to change the label of a particular node.

8.3 Perturbations

In this section, we introduce the text and structural perturbations used throughout our study.

8.3.1 Natural Language Perturbations

As discussed above, existing attacks are not directly designed for text-attributed graphs nor LLM+GNNs. However, directly, perturbing natural language offers an interesting perspective as it is possible to perform semantically consistent perturbations, as one can broadly gauge if the semantics of the text have changed. In contrast, when changing graph structure, as recent work has argued, most changes are in-fact not semantically consistent. In addition to academic interest, perturbing text can be a more natural threat model, where attackers may try to embed keywords or phrases to trigger certain responses; potentially exploiting inherited vulnerabilities of LLMs to prompt injection, backdoor [252] or jailbreaking [253] attacks.

Indeed, it was recently shown that LLMs used in search engine [254] or product recommendation systems [255, 256] can be manipulated to return specific items that may be factually inaccurate or not align with the users interests. For example, [257] recently demonstrated that adding "strategic text sequences" can lead to previously low-ranked products to be highly listed. Given that text-attributed graphs are often found in recommendation settings, such vulnerabilities are particularly concerning as such strategies may be combined with structural perturbations to increased detriment.

Here, we consider both adversarial text attacks and semantics preserving text perturbations, and compare the effects of both strategies on exemplar LLM-As-Predictor and LLM-As-Enhancer methods. We begin by discussing semantic preserving text perturbations.

8.3.2 Semantics Preserving Text Perturbations

While adversarial attacks necessarily seek to generate inputs challenging for the model, often by performing gradient descent against the loss, here, we consider a more benign setting that seeks to understand the sensitivity of models to artifacts of the provided text attributes. Our investigation is motivated by findings that PLMs can be sensitive to mis-spellings, l33t-speak, and synonym substitution amongst other natural modifications. While LLMs display more robustness to such modifications, they are nonetheless susceptible to seemingly unimportant phrasing alterations such as changing the order of multiple choice options or few-shot examples. Indeed, [258] recently argued for performing multi-prompt evaluation, as models exhibit considerable variance when tasks are presented using different prompts. Insofar as *Enhancers* rely upon PLMs to obtain semantic embeddings and *Predictors* rely upon LLMs to output decisions, it is important to understand how such perturbations affect performance and how incorporating structural information, through neighborhood aggregation or tailored prompts, may influence performance.

While the aforementioned strategies are designed to broadly preserve the intent of the un-attacked text, [259] have argued that intent may in fact be harmed during synonym substitution, leading to malformed adversarial examples. This issue may be exacerbated under more challenging natural perturbations. Recognizing this limitation, here, we directly prompt an LLM to create the altered text, taking care to instruct the LLM to avoid altering the semantics or unique features. We consider five such perturbations whose prompts are shown in Table. 8.1. We manually inspected the generated text on a subset of examples to ensure that the semantics were indeed not altered and adjusted the prompt to get the desired

Table 8.1: **Prompts for Generating Natural Perturbations.** Motivated by how users may seek to rephrase their queries, here, we consider five natural perturbations and provide the prompts we used to generate the perturbed text. For readability, the instructions are shown only once but included in all perturbations, while the example is adapted accordingly.

Perturbation	**Prompt Instruction**
Synonym Substitution	Replace key terms with synonyms while preserving meaning. Please avoid altering the primary intent or unique features, and follow the format EXACTLY as shown in the examples when answering, making sure to not provide extra information. This is very important for the response to be parsed correctly. Thanks! [EXAMPLE] Original Text: ”The quick brown fox jumps over the lazy dog.” [EXAMPLE ANSWER] {”modified”: ”The speedy brown fox leaps over the lazy dog.”}
Paraphrasing	Paraphrase the text, preserving primary intent and unique features.
Adding Noise	Insert irrelevant sentences or phrases into the text. Ensure additions are unrelated to main topic.
Truncation	Shorten the text by removing non-essential details; keep the key message clear.
Reordering	Rearrange sentence order while maintaining coherence. Key points should remain clear.

behavior. (See Table. 8.2 for an example.)

8.3.3 Adversarial Text Perturbations

As discussed in Sec 8.2, a variety of adversarial strategies have been proposed to decrease the performance of the PLMs and LLMs. Here, we focus on PLM attacks that perturb the natural language input and use the perturbed input to evaluate both *Enhancers*, after embedding the perturbed input, and *Predictors*, by prompting the LLM with the perturbed input. We leave the exploration of adversarial prompt engineering techniques for future work.

Adversarial PLM attacks can broadly be categorized by granularity of their perturbations, namely, sentence-level, word-level or character-level. Here, we consider four representative PLM attacks, using the OpenAttack library (see Table. 8.3.3), and introduce them briefly. VIPER [260] is a black-box character-level attack that replaces individual characters with those that appear visually similar, e.g., ”o $\rightarrow$ 0.” Though humans are fairly robust to such visual substitutions, PLMs appear to struggle and LLMs have also demonstrated vulnerability to prompt reformatting, including changes around adding spaces or changing cases [261, 258].

Table 8.2: **Natural Perturbation Example on OGBN-Products.**

Perturbation	**Description**
None	Historic Stage Routes of San Diego County is a product of the County of San Diego Department of Parks and Recreation and their Parks Society, many generous historical societies, historians, and people who love the stage routes. Ellen L. Sweet is a local researcher with extensive knowledge of San Diego history. Lynne Newell, PhD, is an archaeologist...
Synonym Substitution	Vintage Stage Routes of San Diego County is a product of the County of San Diego Department of Parks and Recreation and their Parks Association, many charitable historical organizations, scholars, and people who love the stage routes.
Adding Noise	The historic Route 66 is one of the most famous roads in the US. Historic Stage Routes of San Diego County is a product of the County of San Diego Department of Parks and Recreation and their Parks Society, many generous historical societies, historians, and people who love the stage routes...
Truncation	Historic Stage Routes of San Diego County is a product of the County of San Diego Department of Parks and Recreation and their Parks Society, authored by Ellen L. Sweet and Lynne Newell, PhD.
Paraphrasing	San Diego County's Historic Stage Routes is a collaboration between the County of San Diego Department of Parks and Recreation, their Parks Society, various historical societies, historians, and stage route enthusiasts. Local historian Ellen L. Sweet and archaeologist Lynne Newell, PhD, and former county historian...
Reordering	Lynne Newell, PhD, is an archaeologist and the former county historian, and Ellen L. Sweet is a local researcher with extensive knowledge of San Diego history. They are also the authors of Ranchos of San Diego County. Historic Stage Routes of San Diego County is a product of the County...

HotFlip [262] is a word and character-level white box attack that uses a combination of adversarial gradient descent and beam search to perform ”flips”, i.e. character changes, that harm classifier performance. TextFooler [263] is a black-box attack that uses an importance score to identify words that are influential in the classifier's decision and then uses synonym substitution, part-of-speech verification to ensure that the perturbed sentence is semantically consistent. Bert-Attack [264] is a black-box method that first computes the importance scores of different words to select a set of candidates for replacement and then uses BERT (without any task finetuning) to help identify replacements that still respect the surrounding context. In practice, BERT-Attack proven to be a strong baseline.

Table 8.3: **Overview of Adversarial Attack Methods.** Reproduced from [15].

Method	**Type**	**Level**	**Description**
BERT-ATTACK	Score	Word	Greedy contextualized word substitution
HotFlip	Gradient	Word, Char	Gradient-based word or character substitution
TextFooler	Score	Word	Greedy word substitution
VIPER	Blind	Char	Visually similar character substitution

8.3.4 Structural Perturbations

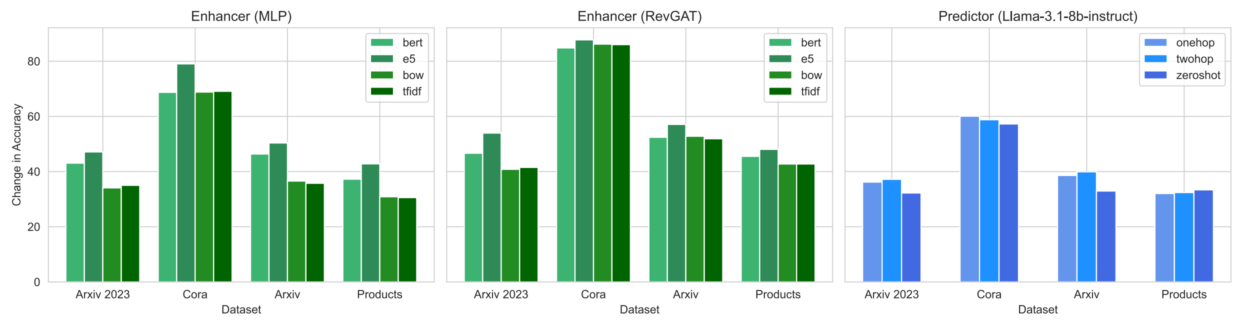
While both feature and topology (structural) attacks have been studied to craft adversarial examples for GNNs, structural attacks are better researched and typically more harmful. This effectiveness has been attributed to several causes, including decreasing homophily, the amplifying effects of message passing and the creation of inconsistent samples [265, 88]. As discussed in Sec. 8.2, a variety of methods have been proposed, with various attack model assumptions. Here, we consider a whitebox setting so that we may evaluate the robustness of Enhancers and Predictors under the strongest attack setting.

To ensure efficiency to large-scale graphs, we utilize the Projected Randomized Block Coordinate Descent (PRBCD) adversarial attack [266], which uses randomized block coordinate descent to avoid simultaneously computing the gradient of $O(n^2)$ potential edge perturbations, and provides strong attacks across graph-size. We perform a targeted evasion attack using PRBCD for *Enhancers* and use the perturbed graph when crafting structurally aware prompts with *Predictors.* In this setting, we are primarily interested in the transferability of GNN-based structural attacks to *Predictors* so we do not consider other attack models. Alternative poisoning and black or gray box settings are left to future work.

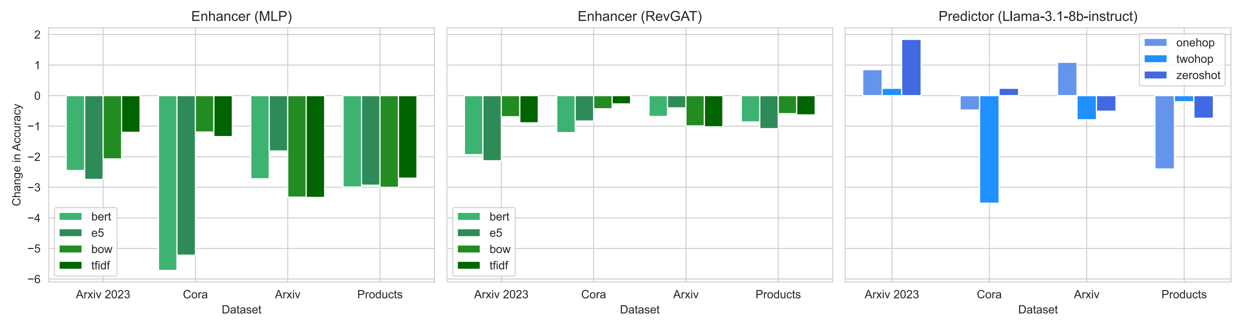
8.4 Experiments

In this section, we evaluate the effects of the structural and text-based perturbations on exemplar LLM-As-Enhancer and LLM-As-Predictor strategies. We note that while there has been growing research in more sophisticated strategies for both these paradigms, we leave them to future work due to the prohibitive computational burden incurred during our comprehensive evaluation.

Experimental Set-up. *Datasets.* All experiments are conducted on four datasets for 5 seeds unless, otherwise noted. The test-set is sub-sampled to 500 samples, such that all classes are equally represented. *Models.* For the baseline GNN performance, we use two types of shallow embeddings, Bag-of-Words (bow) and term frequency-inverse document frequency (tfidf). We further consider 4 different parameterized networks, RevGAT, SAGE, GCN and MLP, as



(a) **Unperturbed Performance.** Results for the MLP and RevGAT-based Enhancer and a Llama-3.1-8b-instruct Predictor. The unperturbed accuracy is typically highest with Enhancer (RevGAT), followed by Enhancer (MLP) and Predictor (Llama). Enhanced embeddings outperform shallow embeddings (bow and tfidf). Although the predictor occasionally surpasses the performance of Enhancer with shallow embeddings, it struggles to be a top performer overall.



(b) **Perturbed Performance.** Average performance over five natural and four adversarial text perturbations. While Predictor struggled to achieve the best performance, it is generally less sensitive to text perturbations, with some exceptions on Cora and Products. The MLP shows the most sensitivity to perturbations, followed by RevGAT. In some cases, Predictor even improves performance.

Figure 8.2: **Aggregated Performance on Text Perturbations.**

structure free option [205]. For the *Enhancer*, we utilize bert-base-uncased and e5-large-v2 as the underlying PLMs from which node attributes are obtained. For the *Predictor*, we use simple structurally aware prompts introduced in [267], which incorporate one-hop or two-hop neighbhors into the prompt. Note that using the entire neighborhood can exhaust the context-length if large, so a fixed number of nodes are randomly sampled from k-hop neighborhood. *llama-3.1-8b-instruct* is used as the predictor. *Textual Perturbations.* We use the OpenAttack library [15] with default settings. A bert-base-uncased model is finetuned for 3 epochs for the surrogate trained PLM classifier. Perturbations are generated for seed and corresponding split. *Natural Perturbations.* Using the prompts discussed in Sec. 8.3.2, we prompt llama-3.1-8b-instruct to generate the perturbed text. *Structural Perturbations.*

Table 8.4: **Predictor Performance on Textual Perturbations.** The change in performance of the *Predictor (llama-3.1-8b-instruct)* with respect to the clean data is shown, where improvements over 1% are shown in green and decreases over 1% are shown in red. We observe that adversarial text-attacks are relatively effective on Cora and Products, but have limited, even positive effect, on Arxiv 2023 and Arxiv. Natural perturbations do not significantly harm performance, with the exception of Cora.

Dataset	**Prompt**	**BertAttack**	**HotFlip**	**TextFooler**	**Viper**	**Noise**	**Paraphrase**	**Reordering**	**Syn. Sub.**	**Trunc.**
Arxiv 2023	onehop	0.38	0.28	-0.40	1.98	1.46	0.00	1.50	-0.58	3.08
	twohop	0.36	-0.52	-0.54	-0.24	0.06	-0.04	0.08	-0.46	3.50
	zeroshot	0.88	2.30	1.16	2.06	2.86	0.74	0.06	0.26	6.28
Cora	onehop	-5.96	-0.85	-3.53	-3.88	-2.76	-1.55	-2.03	-0.58	-0.44
	twohop	-1.18	-2.26	-2.28	-0.56	-1.54	1.00	-0.26	1.46	1.42
	zeroshot	-5.48	-4.56	-5.68	-6.26	-1.14	-1.94	-0.96	-1.78	-3.88
Arxiv	onehop	0.02	-0.30	0.65	-0.28	-0.30	-0.85	0.33	1.45	1.40
	twohop	0.06	0.68	-1.02	-0.35	0.86	-0.62	0.50	-1.06	-0.84
	zeroshot	-0.48	0.82	0.17	2.58	0.98	1.50	-0.13	0.35	4.00
Products	onehop	-1.40	-1.58	-2.28	-0.50	-1.43	-0.20	0.45	-0.70	0.50
	twohop	-1.98	-0.68	-2.78	0.38	-0.68	0.10	0.75	0.88	-0.60
	zeroshot	-0.92	-1.58	-1.35	-0.27	-0.35	-0.85	-0.80	0.05	-0.60

As discussed above, we use the PRBCD attack. The attack strength is set to 5% of the total number of edges. We perform a targeted attack on the test nodes so as to model the strongest attack setting.

Here, we are interested in the effectiveness and transferability of different perturbations with respect to *Enhancers* and *Predictors*. We make the following observations.

Observation 1. Predictors demonstrate less sensitivity to textual perturbations. As shown in Fig. 8.4, *Enhancers* have better performance than the *Predictor* on the unperturbed data and overall. However, when evaluating the average performance on the five natural perturbations and four text perturbations discussed above, we observe that the *Predictor* experiences less loss of performance on most datasets and prompts, even improving performance on the Arxiv 2023 dataset, relative to the unperturbed data. In contrast, we observe that *Enhancer*(MLP) loses the most performance. We hypothesize that the MLP is most prone to overfitting to the training data, and, unlike the RevGAT model, are unable to use neighborhood aggregation to mitigate the effects of the perturbation.

Observation 2. Effectiveness of Individual Text Perturbations on Predictors. While *Predictors* were relatively unaffected by text perturbations, we observe that adversarial attacks do seems to be relatively more detrimental than natural perturbations, especially on the Cora and Products dataset. The advanced reasoning capabilities of LLMs bolsters performance against such perturbations. Notably, there does not seem to be a consistent effect of prompt style on performance, indicating that incorporating additional neighbor information may not be particularly useful in this setting particular attack setting.

Table 8.5: **Enhancer Performance on Text Perturbations.** Results show that all perturbations except reordering reduce performance, with truncation causing the most significant average loss. Adversarial attacks effectively degrade model performance across most settings, though shallow embeddings on Cora and Arxiv 2023 are less affected, possibly due to the limited impact of new characters or words introduced by the attacks using. Further investigation is suggested.

			Adversarial Perturbation				Natural Perturbation				
Model	**Dataset**	**Sentence Model**	**BertAttack**	**HotFlip**	**TextFooler**	**Viper**	**Adding Noise**	**Paraphrasing**	**Reordering**	**Synonym Substitution**	**Truncation**
MLP	**Arxiv 2023**	bert-base-uncased	-2.78	-2.92	-5.30	-1.72	-1.84	-0.66	-0.66	-1.44	-4.72
		e5-large-v2	-3.86	-5.88	-5.98	-0.02	-2.64	-0.50	-0.12	-1.54	-4.12
		bow	-2.53	-0.95	-2.65	-0.95	-0.76	-1.85	-0.53	-2.63	-5.80
		tfidf	0.00	0.00	0.00	0.00	-0.42	-1.74	-0.08	-2.76	-5.84
	cora	bert-base-uncased	-9.80	-7.82	-15.92	-2.50	-4.46	-3.66	0.00	-2.76	-4.54
		e5-large-v2	-12.32	-7.62	-10.06	-2.06	-4.62	-1.80	-0.14	-3.04	-5.32
		bow	0.00	0.00	0.00	0.00	-1.14	-1.34	-0.16	-3.34	-4.70
		tfidf	0.00	0.00	0.00	0.00	-1.64	-1.84	-0.74	-3.86	-4.02
	ogbn-arxiv	bert-base-uncased	-3.51	-4.30	-6.24	-1.72	-1.84	-1.12	-0.25	-0.35	-5.16
		e5-large-v2	-2.83	-4.63	-4.10	-0.23	-2.23	0.10	0.23	-0.10	-2.50
		bow	-5.34	-4.26	-5.96	-6.62	-0.34	-1.02	0.32	-1.44	-5.20
		tfidf	-4.56	-3.98	-5.62	-6.16	-0.78	-1.46	-0.08	-2.16	-5.14
	ogbn-products	bert-base-uncased	-5.08	-4.55	-7.35	-1.25	-4.73	-0.92	-0.85	-0.57	-1.62
		e5-large-v2	-4.98	-6.93	-5.68	-1.30	-4.58	-0.70	-0.30	-0.25	-1.65
		bow	-5.60	-3.13	-7.40	-2.90	-1.53	-1.00	-0.43	-1.33	-3.73
		tfidf	-4.53	-3.38	-7.30	-2.20	-2.08	0.32	-0.28	-1.03	-3.88
RevGAT	**Arxiv 2023**	bert-base-uncased	1.72	-2.16	-3.20	1.04	-1.62	-0.32	-0.62	-1.32	-2.70
		e5-large-v2	-2.12	-1.90	-2.65	0.80	-2.04	-0.42	-0.22	-1.52	-2.28
		bow	1.14	0.00	1.92	1.24	0.00	-1.42	-0.44	-2.14	-3.92
		tfidf	0.00	0.00	0.00	0.00	-1.84	-1.74	-0.04	-2.10	-3.90
	cora	bert-base-uncased	-7.82	-5.62	-8.76	-1.74	-4.42	-0.82	-0.54	-2.54	-4.32
		e5-large-v2	-5.12	-4.34	-6.08	-2.42	-3.68	-1.28	0.22	-2.36	-3.56
		bow	0.00	0.00	0.00	0.00	-1.62	-0.52	-0.24	-2.52	-3.34
		tfidf	0.00	0.00	0.00	0.00	-1.58	-1.22	-0.12	-2.86	-3.46
	ogbn-arxiv	bert-base-uncased	-2.12	-3.14	-4.78	1.36	-2.12	-1.32	-0.18	-0.28	-2.94
		e5-large-v2	-2.54	-3.62	-3.62	0.30	-1.62	0.40	-0.36	-0.08	-2.30
		bow	-4.10	-3.46	-5.02	-4.32	-0.52	-0.72	0.14	-1.22	-4.34
		tfidf	-3.76	-2.82	-4.16	-4.58	-0.62	-1.12	-0.02	-1.96	-3.86
	ogbn-products	bert-base-uncased	-4.20	-3.72	-5.44	0.00	-3.92	-0.40	-0.62	-0.32	-1.26
		e5-large-v2	-4.16	-5.02	-4.38	-0.32	-3.42	-0.28	-0.18	-0.16	-1.08
		bow	-4.90	-2.42	-5.64	-0.92	-0.83	-0.52	-0.48	-1.02	-2.98
		tfidf	-3.96	-2.68	-5.52	-0.62	-2.82	-0.08	-0.12	-1.18	-3.10
GCN	**Arxiv 2023**	bert-base-uncased	-2.94	-3.20	-4.46	-0.96	-2.02	-0.18	0.08	-0.20	-4.52
		e5-large-v2	-4.64	-4.94	-4.96	-0.38	-2.68	-1.44	-0.96	-2.10	-4.10
		bow	-2.34	-0.10	-2.20	-0.10	-0.12	-0.74	-0.46	0.64	-3.14
		tfidf	0.00	0.00	0.00	0.00	-0.74	-2.38	-0.64	-2.64	-6.02
	cora	bert-base-uncased	-2.15	-1.38	-2.63	-0.60	-0.73	-0.45	0.26	-0.98	-0.46
		e5-large-v2	-1.50	-0.87	-1.43	-0.02	-0.59	-0.36	0.10	-0.39	-0.69
		bow	0.00	0.00	0.00	0.00	-0.24	0.17	0.37	-0.23	-0.13
		tfidf	0.00	0.00	0.00	0.00	-0.10	0.88	-0.02	-0.24	0.27
	ogbn-arxiv	bert-base-uncased	-0.40	-0.87	-0.60	0.25	-0.68	-0.10	-0.15	-0.50	-0.55
		e5-large-v2	-0.20	0.00	-0.40	0.00	0.20	0.00	0.70	0.70	-0.40
		bow	-0.58	-0.60	-0.92	-0.82	-0.25	0.17	-0.08	-0.25	-0.20
		tfidf	-0.35	-0.18	0.00	-0.98	-0.18	0.08	-0.05	-0.10	0.12
SAGE	**Arxiv 2023**	bert-base-uncased	1.54	-2.48	-1.90	1.20	-3.10	-1.34	-0.44	-2.40	-4.02
		e5-large-v2	-2.34	-3.26	-4.00	0.70	-2.14	-0.24	-0.38	-2.70	-3.68
		bow	1.02	0.00	1.70	1.22	0.00	-1.36	-0.32	-2.12	-3.80
		tfidf	0.00	0.00	0.00	0.00	-1.62	-1.72	-0.04	-2.30	-4.20
	cora	bert-base-uncased	-7.20	-5.50	-8.40	-2.00	-4.60	-0.90	-0.60	-2.80	-4.20
		e5-large-v2	-5.00	-4.00	-6.20	-2.80	-3.80	-1.40	0.20	-2.50	-3.60
		bow	0.00	0.00	0.00	0.00	-1.54	-0.50	-0.20	-2.40	-3.30
		tfidf	0.00	0.00	0.00	0.00	-1.80	-1.30	-0.10	-2.70	-3.40
	ogbn-arxiv	bert-base-uncased	-2.30	-3.00	-4.60	1.20	-2.20	-1.20	-0.10	-0.20	-3.00
		e5-large-v2	-2.70	-3.50	-3.80	0.20	-1.70	0.20	-0.30	-0.05	-2.60
		bow	-4.40	-3.20	-5.00	-3.60	-0.50	-0.60	0.12	-1.10	-4.20
		tfidf	-3.80	-2.50	-4.00	-4.20	-0.70	-1.00	-0.05	-2.00	-3.80
	ogbn-products	bert-base-uncased	-4.30	-3.90	-5.50	0.00	-4.10	-0.50	-0.70	-0.60	-1.40
		e5-large-v2	-4.00	-5.30	-4.80	-0.40	-3.90	-0.30	-0.20	-0.25	-1.20
		bow	-5.00	-2.80	-6.00	-0.90	-0.90	-0.50	-0.55	-1.50	-3.20
		tfidf	-3.70	-3.00	-5.40	-0.60	-2.90	-0.10	-0.15	-1.70	-3.50

Observation 3. Effectiveness of Individual Text Perturbations on Enhancers. From the results in Table. 8.4, we observe that all but the re-ordering perturbation seem to effect the performance of the Enhancers, across different types of embeddings. In particular,

Table 8.6: **Enhancer Performance on Structural Perturbations.** A targeted PRBCD attack at 5% global budget is performed for *Enhancer* with a GCN backbone and different node featurizations. The unperturbed clean performance and post-attack perturbation performance are reported. As expected there is a significant decrease in performance, however, we do not necessarily see a clear benefit of the semantically aware embeddings (bert/e5) used by *Enhancers.*

Dataset	Sentence Model	Clean Performance	Perturbed Performance	Change
Arxiv 2023	bert-base-uncased	42.38	7.02	-35.36
	e5-large-v2	50.03	11.53	-38.50
	bow	35.74	5.94	-29.80
	tfidf	37.00	5.74	-31.26
Cora	bert-base-uncased	81.02	70.30	-10.72
	e5-large-v2	79.83	69.68	-10.15
	bow	82.65	70.75	-11.90
	tfidf	81.44	70.66	-10.79
Arxiv	bert-base-uncased	40.48	0.00	-40.48
	e5-large-v2	41.70	0.00	-41.70
	bow	30.70	0.58	-30.12
	tfidf	31.38	0.40	-30.98
Products	bert-base-uncased	38.23	9.00	-29.23
	e5-large-v2	40.03	10.44	-29.59
	bow	32.13	7.13	-25.00
	tfidf	33.23	7.10	-26.13

truncation appears to be the most detrimental natural perturbation (based on the average loss of performance). The adversarial attacks are effective against both Enhancer(MLP) and Enhancer(RevGAT), but we note that there is an interesting exception where shallow embeddings are not effected on Cora and Arxiv 2023. We hypothesize that this may be because the attacks introduced a small number characters or words that were not considered in the original corpus and could be effectively ignored, though future investigation is needed.

Observation 4. Effectiveness of Structural Perturbations on Enhancers. As discussed in Sec. 8.2.2, we perform a targeted PRBCD attack and therefore expect a significant decrease in performance. Here, we are mainly interested if the semantically aware embeddings used by Enhancers can help mitigate this decrease. While bert and e5 (the semantically aware embeddings) outperform the shallow embeddings on Arxiv 2023 with respect to minimize the decreased performance and maximizing the final accuracy, we see on Products that semantic embeddings have a bigger decrease in performance but better overall accuracy. On Arxiv, all models effectively collapse, though the starting accuracy of the semantic embeddings was better.

Observation 5. Effectiveness of Structural Perturbations on Predictors. As shown in Table. 8.7, we see that structural attacks are primarily effective on Cora and Products. Indeed, on Arxiv 2023 and Arxiv, we observe that performance can increase relative to the clean accuracy. Furthermore, we do not see that using a particular sentence model leads to more

Table 8.7: **Predictor Performance on Structural Perturbations.** To determine if structural attacks from GNN-based models transfer to *Predictors*, we evaluate the *Predictor* on the attacked graph structured obtained after attacking a GCN trained with different node embeddings. The post-attack and clean accuracy are reported for the one-hop and two-hop prompting styles, while the structure-free prompt's performance on the clean graph is indicated in parenthesis, where improvements over 1% are shown in green and decreases over 1% are shown in red.

Dataset	Prompt	Bert	e5-large-v2	bow	tfidf	Clean
Arxiv 2023 (32.24)	onehop	39.40	37.97	38.10	37.50	35.80
	twohop	38.97	39.50	39.33	39.80	38.17
Cora (57.26)	onehop	60.13	58.97	59.53	59.07	61.53
	twohop	58.20	58.67	58.03	59.87	59.37
Arxiv (32.95)	onehop	39.73	39.75	40.87	40.87	38.15
	twohop	40.43	41.45	42.87	41.30	39.47
Products (33.375)	onehop	31.07	29.87	30.03	29.53	31.87
	twohop	31.87	30.53	30.80	31.80	32.13

transferability when attacking the *Predictor* nor that a particular prompting style is more susceptible to the attack. We suspect that randomly sampling the number of nodes included in the prompt from the neighborhood may help mitigate the effect of structural attacks. With sufficient perturbation budget or low-degree nodes, we suspect that the attacks will transfer more effectively.

Overall, our results indicate that there is considerable scope to better understand how interaction of modalities may lead to transferable or previously unexplored attack avenues. We discuss some broader implications of our analysis and future directions next.

8.5 Conclusion & Discussion

Our analysis highlights the importance of studying the vulnerability of GNN+LLM models on text-attributed graphs as they do exhibit vulnerability to different attack modalities. There are several directions of future work that we believe are worth exploring, and will likely prove even more detrimental than those studied here. First, while this analysis focuses on existing text-only and structure-only attacks, designing attacks that jointly attack both modalities natively are likely to lead to considerable more damage. For example, one may use techniques from structural attacks to select vulnerable nodes, and use the LLM to generate the corresponding adversarially perturbed node attributes. Indeed, we suspect that such joint attacks would be better able to break both structural and feature homophily, which has been shown to harm both GNN adversarial robustness and bolster GNN+LLM performance.

Another interesting line of work is to design backdoor and instruction tuning attacks, where the “trigger” or sensitive words are captured through both the graph structure and text attributes, making it harder to detect such malicious behavior. Indeed, we suspect that more sophisticated LLM+GNN methods, which often perform some instruction or finetuning, may be particular sensitive to such attacks. Overall, our analysis indicates the importance of studying the robustness and sensitivity of these models and believe there are many directions worth pursuing.

CHAPTER 9

Conclusions and Future Work

9.1 Summary

In this thesis, we considered three lenses for improving graph representation learning beyond accuracy. Namely, we (1) rigorously analyzed limitations of popular constrastive learning training protocols, (2) proposed novel methodologies for improving uncertainty estimation of graph-neural network based classifiers, and (3) studied joint models, which combine the world knowledge of large language models with GNNs, on two text-attributed graph tasks.

Augmentations in Graph Contrastive Learning. In Part I of the thesis, we focused on the role augmentations play in graph constrastive learning (CL) through empirical and theoretical analyses. In Chapter 3, we empirically identified several limitations of popular domain agnostic graph augmentations (DAGAs). We reported that such augmentations often destroy task-relevant information, which can lead to deteriorated downstream performance. Indeed, we found that the invariances induced by DAGAs are often irrelevant to downstream tasks. To reconcile the discrepancy between known desirable CL properties and performance, we identified that the inductive bias of randomly initialized GNNs is a bolstering factor. Building upon these insights, we conducted two case studies that demonstrated the benefits of two broad domain aware augmentation strategies: generating augmentations in the abstracted modality or designing augmentations that create invariance to purely task-irrelevant information (to avoid destroying useful information).

Building upon these empirical insights, we analyzed graph constrastive learning through a set of data-centric properties in Chapter 4. We conducted a generalization analysis of graph contrastive learning that decomposed DAGAs as a series of graph edit operations that allowed us to demonstrate there exists a performance-separability trade-off that is related to intrinsic dataset properties. We further introduced a synthesis data generating process that support benchmarking against gold-standard domain-aware augmentations. We found that automated augmentations continue to struggle, even domain-aware augmentations are available for selection, indicating both the utility of the benchmark and the importance of further studying this problem. Overall, this part of the thesis provided novel insights and resources on improving an important training paradigm. Building on the insights and correcting these limitations will enable practitioners to better access the benefits of CL, helping support better representations beyond accuracy.

Uncertainty Estimation with Graph Neural Networks. In Part II, we focused on improving the uncertainty estimation of GNN-based classifiers. Insofar as reliable uncertainty estimation underlies several safety critical tasks of model deployment, such as out-of-distribution detection, and generalization gap prediction, it is a key aspect of improving graph representation learning. In Chapter 5, we proposed novel training protocol, G-ΔUQ, that leds to improved uncertainty estimations, on not only on in-distribution data, but also on covariate or concept shifted data. We demonstrated G-ΔUQ's performance on node classification as well as graph classification tasks. To support pretrained models, we further proposed partially stochastic variants of G-ΔUQ.

To improve the uncertainty estimation of GNNs used for link prediction, we further proposed E-ΔUQ in Chapter 6. This training protocol is designed to incorporate node level uncertainties on head and tail nodes to better represent the overall link uncertainty. We designed three E-ΔUQ variants, with different levels of stochasticity. We found that incorporating a simple node-level pretraining task can help improve calibration performance. In summary, this part of the thesis makes novel methodological contributions to towards safer and more reliable GNNs.

Large Language Models and Graph Representation Learning. The final part of this thesis focuses on graph learning tasks performed on text-attributed graphs, e.g., graphs with natural language node attributes. Combining structural information, through the use of GNNs, and world-knowledge, through the use of LLMs, allows single modality models to overcome their limitations. To this end, in Chapter 7, we proposed GCLR (graph clustering with LLM refinement), an active learning framework for improving the performance of graph clustering on text-attributed graphs. GCLR is designed to avoid the training expenditure incurred by the pre-training, co-training or finetuning of the LLM when infusing world knowledge into the graph-based solution. Furthermore, GCLR is designed to avoid expensive prompting expenditure that would arise when prompting for each node in graph. In particular, we treated the LLM as a noisy oracle in an active learning setting that would only provide feedback on the most challenging nodes to avoid prompting the LLM over all potential nodes.

We demonstrated that GCLR improves performance over LLM-only or GNN-only models across datasets, and is flexible to the choice GNN-based clustering method.

In Chapter 8, we considered the robustness of joint LLM+GNN models. In particular, we studied the susceptibility and transferability of LLM-As-Enhancer and LLM-As-Predictors to natural text perturbations, adversarial attacks, and structural attacks. Our results indicated that both paradigms are sensitive, in varying degrees, to the perturbations. Indeed, we found that LLMs-As-Predictors generally displayed less sensitivity to textual and structural perturbations, but did generally perform more poorly on the unperturbed dataset. We further noted that Enhancers are susceptible to pretrained language model based attacks, and that semantic embedding do not always lead to better performance under adversarial text-perturbations. Overall, this part of the thesis demonstrated the benefits of combining LLMs and GNNs to overcome limitations of either modality and raised several important questions on the vulnerabilities that may be introduced by doing so.

9.2 Future Work

Each part of this thesis offers several exciting directions of future work, considered jointly as well as separately. We discuss some them here.

Using LLMs to Augment Text-Attributed Graphs. As discussed in Part I, though well-designed augmentations are critical for strong downstream performance, it can be challenging to design them. To this end, another promising direction is to leverage the generative capabilities of LLMs to augment text-attributed graphs. Specifically, LLMs could be employed to synthesize realistic but varied node descriptions, or suggest complementary graph structures that may denoise the original graph. Notably, LLMs' world knowledge could help ensure that generated samples preserve semantic content where necessary and capture task relevant information. Such augmentations could further improve the robustness of LLM+GNN models if augmented samples are specifically designed to counter-attack known weaknesses or bolster underrepresented domains. While there has been work on using LLMs to augment text-attributed graphs using explanations, class descriptions and other strategies, there remains considerable and varied scope for other augmentation strategies.

Designing Methods for Uncertainty Quantification When Working with Joint LLM/GNN Models. As discussed in Part II, reliable uncertainty quantification is critical for deploying models in practice. Quantifying uncertainty of joint LLM+GNN models can be challenging as the prediction uncertainty arises from both the LLM and

GNN. This problem is further complicated by the different paradigms and strategies for creating joint models, many of which may not be directly amenable to existing uncertainty estimation methods. Developing novel methods for such models is an interesting area of future work. A complementary line of work, here, includes understanding the explainability and interpretability of the predictions and reasoning provided by joint LLM+GNN models. Furthermore, studying the robustness of these uncertainty estimates and explanations under data perturbations, such as adversarial attacks or distributional shifts, could yield deeper insights into model behavior and help develop adaptive defense and augmentation strategies.

Novel Attacks and Defenses. Though we considered the transferability of existing unimodal text and structural attacks in Chapter 7 and saw that models do display sensitivity to such perturbations, there remains considerable scope for the design of natively multimodal attacks that may lead to considerably more harm, and more challenging detection. For example, backdoor attacks, a well-known vulnerability of LLMs that arises by poisoning unfiltered training data to later activate a trigger, may also be exploited in LLM+GNN models. Indeed, in vision-language multimodal models, it has been shown that images can be used to inject backdoors at test-time [268]. Similarly, in LLM+GNN models, it may be possible to embed the trigger through a combination of structure and text, making the attack much harder to detect, while potentially spreading its influence across the graph.

LLMs have also demonstrated vulnerability to prompt injection attacks, where benign instructions are used to lead to malicious outputs. Given the prevalence of GNNs in recommendation systems and the growing use of LLM+GNN models in this setting, there is impetus to develop prompt injection attacks designed to attack such models. For example, to increase the visibility of their product or surpass filtering of inappropriate products, sellers may be motivated to modify text-attributes of their products with injected prompts. The effects of such prompts may be amplified if the modified products are strategically placed, and act as decoys for the target product. Overall, incorporating GNNs and LLMs offers avenues for new attacks that supercharge the vulnerability of LLMs by propogating their harms throughout the graph, hide their malicious intent through structure.

APPENDIX A

Better Practices in Graph Contrastive Learning

A.1 Experimental Details of Section 3

For Secs. 3.3.2, 3.3.3 experiments, we use a GIN-based encoder [27] similar to InfoGraph [14] and GraphCL [1] for all datasets but (DEEZER, GOSSIPCOP, GITHUB-SGZR). PNA is used for (DEEZER,GITHUB-SGZR) to stabilize Infograph's loss and in Sec. 3.3.1. For GOSSIPCOP, the encoder is based off PyG's implementation [269]: 1 GCN Layer, 1 Linear Layer, embedding dimension = 128, Optimizer = Adam [146], LR = 0.001, # of Epochs = 25, batch size = 128.

Sec. 3.3.1 Experimental Setup: The following training configuration is used: # of Layers = 3, LR = 0.01, # of Epochs = 30, Batch-Size = 32. Models are trained on a Nvidia Tesla K80 GPU with Adam. A batch-norm layer is included between the output of the backbone and cross entropy layer. For augmentations, we follow [1] and stochastically apply node dropping at 20% of graph size and subgraph dropping at 20% of graph size.

Sec. 3.3.2 Experimental Setup: 3-layer GIN model with hidden dimension, learning rate, and epochs trained of (32, NA, NA) for RAND (Random Initialization), (512,0.001,20) for InfoGraph, and (32,0.01,20) for GraphCL. Adam and Nvidia Tesla K80 GPUs (12-GB GPU) were used to train all models. Results for MVGRL ([51]) are not included as we consistently witnessed Out-Of-Memory errors. Results are reported over 3 seeds. *Additional Results:* Fig. A.1a includes additional results for PROTEINS, NCI1 and DD datasets.

Sec. 3.3.3 Experimental Setup: For all datasets, excluding DEEZER and GITHUB-SGZR, we report results from GraphCL and InfoGraph. We use the same GIN encoder as GraphCL when reporting the performance of randomly initialized models for these datasets. On GITHUB-SGZRS, InfoGraph training time on exceeds eights hours using a NVIDIA Tesla P100. *Additional Results:* See Table A.1. We find that the inductive bias of GNNs is strong across different architectures (GraphSAGE, PNA, GCN, and GAT).

Table A.1: **Inductive Bias**: Additional results.

GraphSAGE	3 Layer	4 Layer	5 Layer	GraphCL	InfoGraph
MUTAG	0.85 ± 0.005	0.85 ± 0.006	0.85 ± 0.005	0.82 ± 0.040	0.85 ± 0.005
PROTEINS	0.73 ± 0.004	0.73 ± 0.003	0.74 ± 0.005	0.75 ± 0.002	0.74 ± 0.008
NCI1	0.74 ± 0.003	0.75 ± 0.006	0.73 ± 0.011	0.78 ± 0.000	0.79 ± 0.002
DD	0.77 ± 0.006	0.78 ± 0.002	0.78 ± 0.005	0.80 ± 0.008	0.77 ± 0.010
REDDIT-B	0.85 ± 0.014	0.83 ± 0.016	0.83 ± 0.005	–	0.66 ± 0.137
IMDB-B	0.66 ± 0.012	0.81 ± 0.008	0.81 ± 0.008	–	–
PNA	3 Layer	4 Layer	5 Layer	GraphCL	InfoGraph
MUTAG	0.88 ± 0.011	0.88 ± 0.010	0.89 ± 0.009	0.86 ± 0.023	0.90 ± 0.014
PROTEINS	0.74 ± 0.003	0.74 ± 0.012	0.74 ± 0.005	0.74 ± 0.007	0.74 ± 0.003
NCI1	0.67 ± 0.008	0.68 ± 0.011	0.68 ± 0.010	0.78 ± 0.008	0.77 ± 0.019
DD	0.76 ± 0.014	0.76 ± 0.002	0.76 ± 0.008	0.80 ± 0.008	0.76 ± 0.006
REDDIT-B	0.90 ± 0.003	0.88 ± 0.014	0.89 ± 0.010	0.92 ± 0.006	0.92 ± 0.006
IMDB-B	0.72 ± 0.007	0.68 ± 0.011	0.68 ± 0.010	0.71 ± 0.009	0.71 ± 0.009
GCN	3 Layer	4 Layer	5 Layer	GraphCL	InfoGraph
MUTAG	0.85 ± 0.003	0.85 ± 0.004	0.85 ± 0.005	0.82 ± 0.013	0.85 ± 0.003
PROTEINS	0.74 ± 0.003	0.73 ± 0.007	0.74 ± 0.004	0.75 ± 0.004	0.75 ± 0.003
NCI1	0.76 ± 0.004	0.75 ± 0.001	0.75 ± 0.002	0.78 ± 0.008	0.79 ± 0.007
DD	0.78 ± 0.002	0.77 ± 0.012	0.78 ± 0.003	0.79 ± 0.007	0.76 ± 0.003
REDDIT-B	0.52 ± 0.005	0.51 ± 0.003	0.52 ± 0.005	0.92 ± 0.002	0.80 ± 0.062
IMDB-B	0.54 ± 0.001	0.57 ± 0.016	0.58 ± 0.008	0.71 ± 0.011	0.62 ± 0.070
GAT	3 Layer	4 Layer	5 Layer	GraphCL	InfoGraph
MUTAG	0.84 ± 0.003	0.85 ± 0.009	0.84 ± 0.003	0.81 ± 0.032	0.85 ± 0.013
PROTEINS	0.74 ± 0.002	0.74 ± 0.005	0.74 ± 0.006	0.74 ± 0.007	0.74 ± 0.005
NCI1	0.76 ± 0.009	0.75 ± 0.004	0.76 ± 0.002	0.78 ± 0.004	0.70 ± 0.040
DD	0.78 ± 0.005	0.77 ± 0.006	0.79 ± 0.001	0.79 ± 0.003	0.76 ± 0.005
REDDIT-B	0.52 ± 0.005	0.53 ± 0.004	0.52 ± 0.012	0.75 ± 0.004	–
IMDB-B	0.51 ± 0.004	0.51 ± 0.009	0.50 ± 0.005	0.51 ± 0.007	–

A.2 Document Classification

In Sec. 3.4.1, we demonstrate the benefits of using task-aware augmentations on a graph-based document classification task.

Experimental Setup: We use the model, code base and default settings of [2]. Models are

Table A.2: **Document Classification**: We use the same augmentations as in Table 3.4. Text-to-Graph augmentations perform synonym replacement as modifying node features.

Augmentation	**(SimSiam) KNN Acc.**	**(BYOL) KNN Acc.**
S. vs S. (ws = 2)	62.62 ± 3.21	66.25 ± 2.65
S. vs N. (ws = 2)	57.35 ± 2.47	62.83 ± 2.82
Text-Space (ws = 2)	$\mathbf{83.69 \pm 0.01}$	$\mathbf{82.69 \pm 1.98}$
Text-to-Graph (ws = 2)	$\mathbf{83.33 \pm 1.29}$	78.16 ± 2.11
S. vs S. (ws = 4)	63.70 ± 8.71	67.53 ± 5.00
S. vs N. (ws = 4)	54.77 ± 1.42	65.99 ± 2.78
Text-Space (ws = 4)	$\mathbf{83.29 \pm 0.9}$	$\mathbf{72.91 \pm 4.97}$
Text-to-Graph Space (ws = 4)	$\mathbf{84.67 \pm 1.57}$	$\mathbf{77.96 \pm 2.04}$

Table A.3: **Comparison to [16]**. Results only reported for SimCLR, as it performs better than SimSiam and BYOL in preceding experiments.

Rand Init.	**ND (20%)**	**ND (30%)**	**Colorize**	**DACL [16]**
37.79 ± 0.03	68.56 ± 0.16	68.07 ± 0.37	$\mathbf{73.67 \pm 0.10}$	59.94 ± 0.01

trained using Adam: lr = 0.001, weight-decay = 1e-4 and cosine scheduler (T=8). We use the code (https://github.com/ jasonwei20/eda-nlp) and augmentations by [5]. Synonym replacement, random deletion, random insertion and random swapping are applied at $5\%, 10\%, 5\%, 5\%$ of sentence length respectively. We generate an augmented version of each sentence for every training epoch. For domain agnostic augmentations, we apply random node dropping (10%) to generate one view. The other view is generated by applying random node or subgraph dropping (10%).

As noted in Sec. 3.4.1, natural language augmentations can be directly in graph space. We provide proof of concept using the synonym replacement augmentation. In Table A.2, results are reported for a model trained with synonym replacement and graph space equivalent, node replacement at 5%. This model achieves comparable accuracy to the original task-aware augmentations. We suspect that synonym replacement is crucial for this task.

A.3 Super-pixel Classification

In Sec. 3.4.2, we demonstrate the benefits of using task-aware augmentations via a case study on MNIST superpixel classification.

Experimental Setup: 50K images are used for training, 10K for validation, and 10K for testing. We follow the same procedure as [86] to convert images to superpixel graphs: SLIC

Table A.4: **Super-pixel, Rep. Similarity.** Avg. intraclass and interclass cosine similarity is reported. Colorizing produces representations with the largest difference between intra- vs. inter- class similarity, indicating that representations are well-separated.

Method	**Aug.**	**Intra. Sim**	**Inter Sim.**	**Abs. Diff**	**Rel. Diff**	**Acc.**
SimCLR	ND (20%)	86.671	78.622	8.04	0.0928	68.56 ± 0.16
SimCLR	ND (30%)	87.03	79.05	7.987	0.091	68.07 ± 0.37
SimCLR	Colorizing	80.801	67.812	12.988	0.1607	$\mathbf{73.67 \pm 0.10}$

Table A.5: **Super-pixel Affinity.** Supervised, clean train accuracy is 90.01% and clean test accuracy is 88.69%.

Aug.	**Aug. Train Acc.**	**Aug. Test Acc.**
ND (20%)	39.42 ± 0.011	40.29 ± 0.054
ND (30%)	29.19 ± 0.01	29.09 ± 0.036
Colorizing	$\mathbf{47.86 \pm 0.05}$	$\mathbf{48.97 \pm 0.03}$

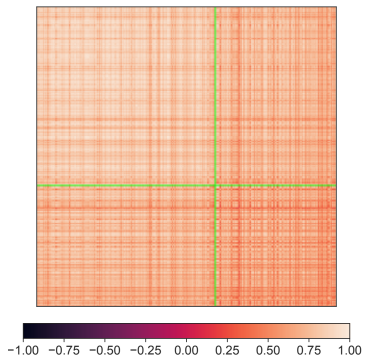
([270]) is used to extract superpixels from the image. Then, a kNN graph is constructed between the superpixels. Node features are RGB values and (x, y) coordinates of superpixels. Classification is performed using three CL frameworks: SimSiam ([121]), SimCLR ([54]), and BYOL ([122]). The same hyper-parameters and architecture are used for all frameworks. Specifically, we use a 5-Layer GIN model closely following [86]. This model is converted from DGL (https://www.dgl.ai) to PyG ([269]). The following hyper-parameters are used: LR=5e-4, Hidden-Dim =110, Epochs=80, Batch-size = 128. The Adam ([146]) Optimizer is used for training. The projector is a 2-layer MLP. The predictor is a 2-layer MLP. Predictor hidden dimension is 1028. Bottleneck dimension is 128. Results are reported over 3 seeds. DAGAs are random node dropping (at 20% and 30%). The task-aware augmentation is random colorizing, performed using Scikit-Image ([271]). As discussed in the main text, colorizing can be represented as transformation on node features as well.

Additional Results: [16] proposes to mix-up samples at either the input or hidden representation level as an alternative to domain-specific augmentations. However, we find that [16] under-performs both node-dropping and colorizing, despite tuning the mixing parameter, α (see Table. A.3).This indicates that context-aware and topological augmentations are still important to GCL. Table A.4 shows intra/inter similarity and Table A.5 shows the affinity.

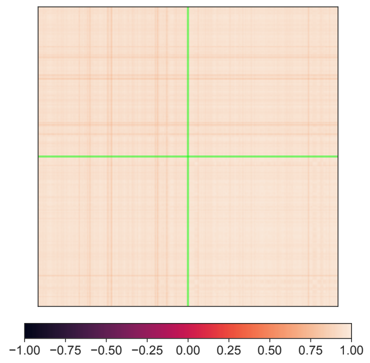
A.4 Additional Related Work

Graph Data Augmentation. [113] train a neural edge predictor to increase homophily by adding edges between nodes expected to be of the same class and break edges between nodes of expected dissimilar classes. However, this approach is expensive and not applicable to graph classification. [112] focus on feature augmentations because it is easier than designing information preserving topological transformations. They add adversarial perturbations to node features as augmentations. In unsupervised settings, labels are not available and cannot be used for the adversarial perturbation, so the proposed approach is not directly applicable. Since the writing of this paper, several recent works have been proposed that perform automatic data-augmentation, some of which we briefly describe in Table B.7.

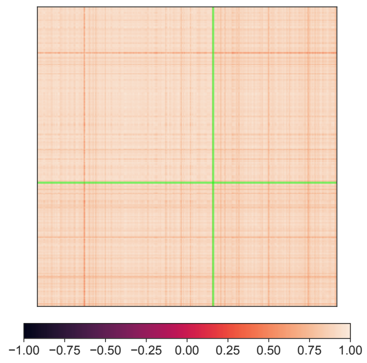
Graph Self-Supervised Learning. Several paradigms for self-super-vised learning in graphs have been recently explored, including the use of pre-text tasks, multi-tasks, and unsupervised learning. See [273] for an up-to-date survey. Graph pre-text tasks are often reminiscent of image in-painting tasks [274], and seek to complete masked graphs and/or node features ([275, 95]). Other successful approaches include predicting graph level or property level properties during pre-training or part of regular training to prevent overfitting ([95]). These tasks often must be carefully selected to avoid negative transfer between tasks. Many unsupervised approaches have also been proposed. [14, 276] draw inspiration from [123] and maximize the mutual information between global and local representations; MVGRL ([51]) contrasts different views at multiple granularities similar to [114]; [1, 272, 149, 52, 131] use augmentations to generate views for contrastive learning. See Table B.7 for a summary of the augmentations used.



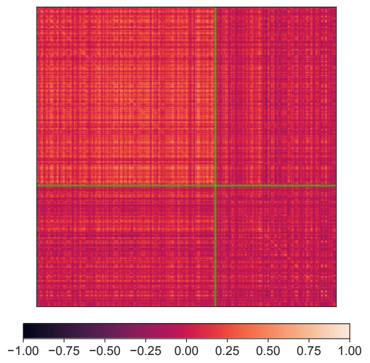
(a) RAND: PROT. (73.678 ± 6.91)



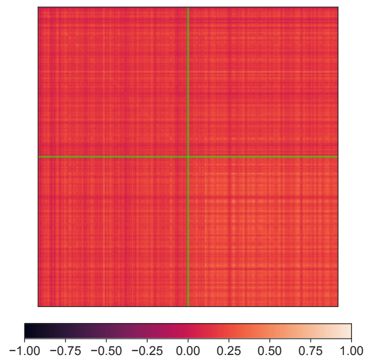
(b) RAND: NCI1 (70.65 ± 1.99)



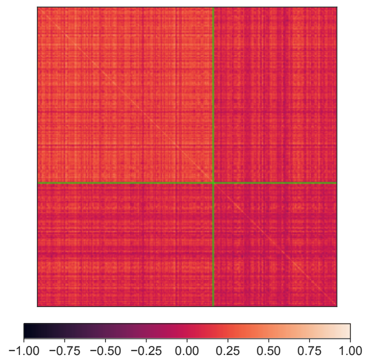
(c) RAND: DD (74.52 ± 9.12)



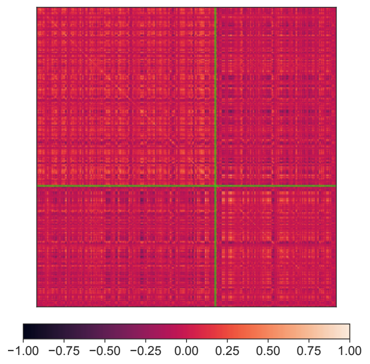
(d) GraphCL: PROT. (73.49 ± 0.33)



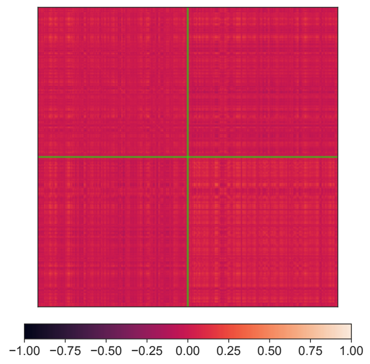
(e) GraphCL: NCI1 (78.16 ± 0.51)



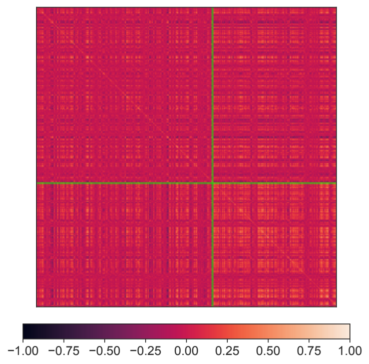
(f) GraphCL: DD (79.54 ± 0.698)



(g) InfoGr: PROT. (73.225 ± 0.36)



(h) InfoGraph: NCI1 (73.58 ± 0.16)



(i) InfoGraph: DD (69.41 ± 0.58)

Figure A.1: **Representational Similarity.** In addition to MUTAG (Figure 3.2), we provide results on PROTEINS, NCI1 and DD. Random inductive bias is most noticeable on MUTAG and PROTEINS. Note that the intra-class similarity can be low for GraphCL and InfoGraph.

Table A.6: **Selected GCL Frameworks**

Method	Augmentations
BGRL [52]	Edge Dropping, Attr. Masking
GCA [149]	Edge Dropping, Attr. Masking (both weighted by centrality)
GCC [272]	RWR Subgraph Extraction of Ego Network
GraphCL [1]	Node Dropping, Edge Adding/Dropping, Attr. Masking, Subgraph Extraction
MVGRL [51]	PPR Diffusion + Sampling
SelfGNN [131]	Attr. Splitting, Attr. Standardization + Scaling, Local Degree Profile, Paste + Local Degree Profile
JOAO [97]	Min-Max Optimization to adaptively and dynamically select from DAGA set
GraphSurgeon [130]	Learnable Feature Augmentors that can be applied pre/-post encoding
BYOV [128]	Uses graph generation (regularized by InfoMin + InfoBottleNeck) as viewmaker
AdvGCL [117]	Adversarial/MinMax Optimization over learnable augmentations
AF-GRL [132]	Finds node-level positive samples sharing "local structure and global semantics"
LG2AR [129]	Learns a policy over augmentations and their respective strengths without bi-level optimization

APPENDIX B

Data-Centric Analysis of Graph Contrastive Learning

B.1 Extending our Analysis to other Loss Functions

While our analysis focuses on the spectral contrastive loss (SpecLoss) [7] for ease of exposition, it can also be extended to other contrastive loss functions and predictive methods, such as BYOL [122]. As we noted in Sec. 4.2, this can be easily accomplished by leveraging our insights on representing graph augmentations through composable graph-edit operations and extending the analyses of Saunshi et al. [159] or Wei et al. [150].

Specifically, the contemporary work of Saunshi et al. proposes a general analysis of contrastive loss functionals and yields a generalization bound similar to Thm. 6, e.g., a bound that is dependent on similar data-centric properties and assumptions. In Sec. 4.3, we decompose GGAs using GED, and then derive expressions for data-centric properties, such as partition dissimilarity, using this decomposition. Since the focus of our analysis is on understanding these data-centric properties in terms of intrinsic dataset attributes (e.g., GED between samples), our theory is complementary to the strategy used by Saunshi et al. Indeed, SpecLoss can be replaced with an alternative contrastive loss functional and adapting the analysis conducted in Sec. 4.3, we can extend our results to other contrastive losses. For predictive methods, we can leverage recent work by Wei et al. [150] which provides an analysis for unsupervised learning methods for continuous data domains (such as images) by enforcing representation consistency on augmented samples–i.e., BYOL-like methods. Critically, Wei et al.'s generalization analysis relies on properties of the data-generating process's latent space and makes analogous assumptions to the unified recoverability plus separability assumption used in our own work. Thus, our theoretical analysis can be extended to BYOL-like methods by deriving equivalent analytical expressions for the latent-space properties used by Wei et al. Moreover, by representing GGAs using graph edit operations, our derivation of such properties relies upon minimal assumptions and is straight-forward. We do note, however, that Wei et al. assume that the dimension of learned representations is equivalent to the number of classes in the dataset. This can be an invalid assumption in unsupervised learning. In contrast, our analysis is more flexible since we only assume the latent dimension is greater than the number of classes.

B.2 Evaluation on a Non-Synthetic Dataset

Our analysis in Sec. 4.3 motivates the need for content-aware augmentations (CAAs) by demonstrating that generic graph augmentations (GGAs) often lead to inconsistent samples, harming representation separability and yielding task *irr*elevant invariances. In Sec. 4.4.2, we empirically validated these claims in a controlled setting through our new synthetic benchmark and the corresponding oracle CAAs (see Fig. 4.5). To demonstrate the generality of our analysis in a practical setup, we repeat this experiment in a realistic setting where domain knowledge is available to design content-aware augmentations.

Experimental Setup. We analyze BACE, a molecule-protein interaction dataset. We train our models by closely following the setup of Sun et al. [11], who propose biochemistry-inspired augmentations for learning domain-informed representations. In our paper's terminology, these augmentations can be regarded as content-aware augmentations. To ensure fair comparison, we use only "local" CAA, which does not incorporate additional "global" domain knowledge (see Sun et al. [11] for further details). We compare against the strongest GGA baseline reported by the authors, called "mask edge features" augmentation.

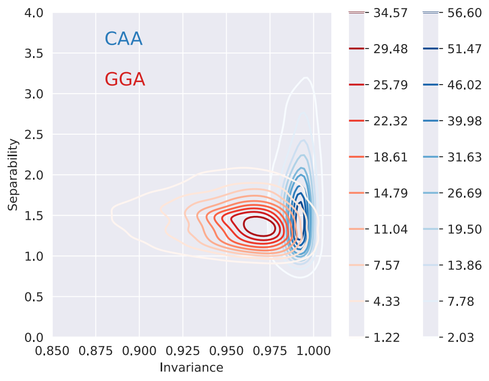


Figure B.1: **Invariance vs. Separability**. On BACE [10], a molecule-protein interaction dataset, we compare the content-aware biochemistry-inspired augmentations from MoCL [11] against the GGAs. In this real-world setting, we see that CAAs induce better invariance and separability (Contours are not filled to improve legibility).

For evaluation, we use the trained models to compute the invariance and separability for each sample. As in Sec. 4.4.2.3, an invariance score is obtained by computing the mean cosine similarity of a sample's representation with 30 of its augmentations. A separability score is computed by dividing the maximum cosine similarity of a given sample and same-class samples by the maximum cosine similarity of a given sample and different-class samples.

Results. As demonstrated in Fig. B.1, the biochemistry-inspired content-aware augmentations induce much better invariance and separability than the GGA. These results provide further corroboration to our synthetic dataset experiments in 4.5) and theory in Sec. 4.3, where we argued that preserving content improves recoverability and leads to task-relevant invariances with better separability.

B.3 On Using Mutual Information for Analyzing Task-Relevance in Augmentations

While several different perspectives have been recently proposed for studying self-supervised learning's behavior, many of these frameworks assume that augmentations induce invariance to information that is *irr*elevant to the downstream task, ignoring the potential for augmentations to induce invariance to task-relevant information and harm generalization performance. However, as we discussed in Sec. 4.2, a notable exception is the information-theoretic analysis of Tian et al. [99]. Specifically, Tian et al. rely upon an information-theoretic framework that interprets the InfoNCE loss as a lower bound of mutual information between two samples. They demonstrate under this framework that optimal augmentations are ones that maximally perturb information irrelevant to the downstream task. However, this viewpoint suffers from the fallacy that InfoNCE is rarely empirically correlated with mutual information. Indeed, Poole et al.[104] demonstrate that this interpretation is only valid when mutual information between two samples is *very* large. For high-dimensional inputs, this will hold true when an augmentation does not alter the input at all, which does not align with the practical behavior of graph (or even image) augmentations. This renders the analysis by Tian et al. relatively inexact compared to our own analysis.

In contrast, we emphasize that our analysis, which has been designed from the ground-up for graph data and augmentations, is more exact. By representing graph augmentations as composable graph-edit distance (GED) operations, we are able to rigorously relate the generalization abilities of a contrastive trained model to intrinsic dataset properties. Specifically, by deriving definitions for partition dissimilarity (Defn 3.8) and inconsistent samples (Lemma 3.6) using GED, our generalization bound relies upon minimal additional assumptions (Thm 6). In Sec. 4.4.2.3 and Sec. B.2, we verify that our theoretical observations are well supported by our experiments on both synthetic and real-world datasets, further demonstrating the validity of our chosen analysis framework.

B.4 Generic Graph Augmentations and Graph Edit Distance

The key insight for our analysis in Sec. 4.3 is that GGAs can be instantiated in a general manner as a composition of graph edit operations. This allows us to derive a unifying assumption related to recoverability and separability in terms of the graph edit distance (GED) between samples. Here, we provide proofs and additional discussion for the statements made in Sec. 4.3. We also discuss how our analysis can be interpreted with respect to the population augmentation graph (PAG) proposed by HaoChen et al. [7].

Table B.1: **Notations**

Symbol	**Definition**
$\overline{\mathcal{X}}$	The original or natural dataset.
$\mathcal{X}$	Set of all augmented data.
$\overline{\boldsymbol{g}} \in \overline{\mathcal{X}}$	Natural (attributed) graph sample.
$\boldsymbol{g}, \boldsymbol{g}' \in \mathcal{X}$	Augmented (attributed) graph samples
$\mathcal{E}_{\overline{\boldsymbol{g}}}$	Edge set of $\overline{\boldsymbol{g}}$.
$\mathcal{V}_{\overline{\boldsymbol{g}}}$	Node set of $\overline{\boldsymbol{g}}$.
$\gamma \in [0,1]$	Augmentation strength. Controls the % of edges or nodes that may be perturbed by the selected augmentation.
$\mathcal{A}(\overline{\boldsymbol{g}})$	The set of augmented samples that can be generated from Augmentation, $\mathcal{A}$, given natural sample $\overline{\boldsymbol{g}}$ and γ.
$\mathcal{A}(\cdot\|\overline{\boldsymbol{g}})$	Distribution of augmentations given a natural sample, $\overline{\boldsymbol{g}}$.
$\mathcal{A}(\boldsymbol{g}\|\overline{\boldsymbol{g}})$	Probability of generating $\boldsymbol{g}$ from $\overline{\boldsymbol{g}}$ given augmentation $\mathcal{A}$.
f	Representation Encoder, $f : \{\overline{\mathcal{X}}, \mathcal{X}\} \to \mathbb{R}^d$
h	Classifier, $h : \mathbb{R}^d \to y$

B.4.1 GGA and Graph Edit Distance

Graph edit distance (GED) is used to capture similarity between two graphs. Intuitively, it captures the cost of making elementary edit operations on a graph, g_1, to transform it to be isomorphic to another graph, g_2. Formally,

Definition 7 (Graph Edit Distance (Defn. 3.1))**.** *Let the elementary graph operators (node insertion, node deletion, edge deletion, edge addition), and the categorical feature replacement operator comprise the set of graph edits. Then,* $GED\left(g_1, g_2\right) = \min_{(e_1,\ldots,e_k)\in\mathcal{P}(g_1,g_2)} \sum_{i=1}^{k} c\left(e_i\right)$*, where* $\mathcal{P}\left(g_1, g_2\right)$ *is the set of paths (series of edit operations) that transforms* g_1 *to be isomorphic to* g_2*. Here,* e_i *is i-th edit operation in the path, and* $c(e_i)$ *is the cost for performing the edit.*

As shown in Table. 4.1, elementary graph edit operators can be used to straight-forwardly represent the *node dropping, edge perturbation and sub-graph sampling* generic graph augmentations [1]. By introducing an additional graph operator, *categorical feature replacement*, we are also able to consider distance with respect to categorical node attributes. This operator performs a "replacement" whenever there is a disagreement between g_1 and g_2's node attributes. Then, the GED is the total cost of structural changes and attribute disagreements between two graphs. Here, we assign a unit cost per operation so all operations are treated equally. Assigning cost to reflect different inductive biases over augmentations is an interesting direction left for future work. Next, we briefly discuss some examples of using graph edit operators to represent GGAs.

Table B.2: **Generic Graph Augmentations vs. Graph Edit Operators. (Reproduced. Table 1.)** GGA can be straightforwardly expressed using graph edit operators.

Augmentations	**Graph Edit Operators**
Node Dropping	Node Deletion
Edge Perturbation	Edge Deletion, Edge Addition
Categorical Attribute Masking	Categorical Feature Replacement Operator
Sub-graph Sampling	Node Deletions

Let $(\overline{\boldsymbol{g}}, \boldsymbol{g})$ represent the original and augmented graph respectively, where we perform *node dropping* to obtain $\boldsymbol{g}$. Recall that the *node dropping* augmentation may only drop up to some fraction of nodes in $\overline{\boldsymbol{g}}$. Then, clearly the minimum cost path can then be found using only *node deletion* operators, and the $GED(\overline{\boldsymbol{g}}, \boldsymbol{g})$ is bounded by the number of allowed node drops. Similarly, if $\boldsymbol{g}$ was obtained through the *edge perturbation* augmentation, which randomly adds or removes a fraction of edges, then $GED(\overline{\boldsymbol{g}}, \boldsymbol{g})$ is bounded by the number of allowable edge modifications and can be obtained using only *edge addition/deletion* operators. (Here, we allow nodes without edges to still exist, so performing node addition/deletion would not result in a lesser *GED*.) The *sub-graph sampling* augmentation extracts a connected sub-graph that contains at most a fraction of total nodes. The minimum cost path can then be defined using only *node deletions*, e.g. where the operator is applied to all nodes not in the sampled sub-graph. Therefore, $GED(\overline{\boldsymbol{g}}, \boldsymbol{g})$ is bounded by $|\overline{\boldsymbol{g}}| - |\boldsymbol{g}|$. As discussed above, the *categorical attribute masking* augmentation can be recovered by directly applying the categorical feature replacement operator. Then, the minimum cost path is then the number of differences between the augmented and original samples' node attributes. We formalize the relationships between augmentations and GED in the following Lemmas.

Lemma 3. ***Allowable augmentations can be expressed using GED. (Reproduction of Lemma 3.2)*** *Let $\overline{\boldsymbol{g}}$ be a natural sample in $\overline{\mathcal{X}}$, $\mathcal{A}$ be some GGA, $\boldsymbol{g} \sim \mathcal{A}(\cdot|\overline{\boldsymbol{g}})$ be an augmented sample generated from $\overline{\boldsymbol{g}}$ and γ be the augmentation strength or the fraction of the graph that GGAs may modify. Then, $\delta \in \{\lfloor\gamma|\mathcal{V}_{\overline{\boldsymbol{g}}}|\rfloor, \lfloor\gamma|\mathcal{E}_{\overline{\boldsymbol{g}}}|\rfloor\}$ represents the number of discrete, allowable modifications for the specified GGA, so $GED(\overline{\boldsymbol{g}}, \boldsymbol{g}) \leq \delta$. Correspondingly, we have $\boldsymbol{g} \in \mathcal{A}(\overline{\boldsymbol{g}}) \Leftrightarrow GED(\boldsymbol{g}, \overline{\boldsymbol{g}}) \leq \delta$.*

Proof. Let $\mathcal{P}$ be the shortest path comprised of the edit operators defined in Table. 4.1 for the given GGA, $\mathcal{A}$. Then, given that at most δ discrete modifications are permitted and each operator has unit cost, $\text{len}(\mathcal{P}) \leq \delta$ and $\sum_{e_i \in \mathcal{P}} c(e_i) \leq \delta$. Thus, $GED(\overline{\boldsymbol{g}}, \boldsymbol{g}) \leq \delta$. $\square$

Lemma 4. *Upper-bound on Size of Augmentation Set.* *The size of $A(\overline{\boldsymbol{g}})$ can be upper-bounded through a combinatorial counting process. For example, to determine $A(\overline{\boldsymbol{g}})$ when the considered augmentation is node dropping, we can delineate all sets of possible nodes with size up-to $\gamma|\mathcal{V}_{\overline{\boldsymbol{g}}}|$. Formally, the upper-bound on the number of samples generated using node dropping are:*

$$|\mathcal{A}(\overline{\boldsymbol{g}})| \leq \sum_{j=1}^{\gamma|\mathcal{V}_{\overline{\boldsymbol{g}}}|} \frac{|\mathcal{V}_{\overline{\boldsymbol{g}}}|!}{(|\mathcal{V}_{\overline{\boldsymbol{g}}}| - j)!j!}$$

We note that this value is an upper-bound because isomorphic pairs are treated as two separate graphs. Furthermore, note the size of the augmentation set grows exponentially with graph size. A similar counting process can be used to determine the number of possible augmented samples obtained through edge perturbation, sub-graph sampling or feature masking. For example, the edge-dropping augmentation could be counted as: $|\mathcal{A}(\overline{\boldsymbol{g}})| \leq \sum_{j=1}^{|\gamma\mathcal{E}_{\overline{\boldsymbol{g}}}|} \frac{|\mathcal{E}_{\overline{\boldsymbol{g}}}|!}{(|\mathcal{E}_{\overline{\boldsymbol{g}}}|-j)!j!}$.

We further note that because generic graph augmentations (GGAs) perturb the graph randomly, each augmented sample, $\boldsymbol{g} \in \mathcal{A}(\overline{\boldsymbol{g}})$, is equally likely, e.g., $\mathcal{A}(\boldsymbol{g}|\overline{\boldsymbol{g}}) = \frac{1}{|\mathcal{A}|}$.

B.5 Details for Generalization Analysis

B.5.1 Generalization Analysis

Recently, HaoChen et al. [7] demonstrated that spectral clustering over a graph that captures similarity of augmented data can recover class partitions as augmentations belonging to the same class are more similar, and thus well-connected. These well-aligned partitions can be recovered through spectral decomposition of the similarity graph and the resulting embeddings can be used as features for downstream tasks. The SpecLoss objective, which performs this decomposition, is then defined as follows [7]: Let $\boldsymbol{g} \sim \mathcal{A}(\cdot|\overline{\boldsymbol{g}})$, $\boldsymbol{g}^{+} \sim \mathcal{A}(\cdot|\overline{\boldsymbol{g}})$, given $\overline{\boldsymbol{g}} \in \overline{\mathcal{X}}$ and $\boldsymbol{g}^{-} \sim \mathcal{A}(\cdot|\overline{\boldsymbol{g}}')$, given $\overline{x}' \sim \mathcal{P}_{\overline{\mathcal{X}}} \wedge \overline{\boldsymbol{g}}' \neq \overline{\boldsymbol{g}}$. Then, for the positive/negative pairs $(\boldsymbol{g}, \boldsymbol{g}^{+})/(\boldsymbol{g}, \boldsymbol{g}^{-})$, the loss $\mathcal{L}(f)$ is:

$$-2 \cdot \mathbb{E}_{\boldsymbol{g},\boldsymbol{g}^{+}}\left[f(\boldsymbol{g})^{\top} f(\boldsymbol{g}^{+})\right] + \mathbb{E}_{\boldsymbol{g},\boldsymbol{g}^{-}}\left[\left(f(\boldsymbol{g})^{\top} f(\boldsymbol{g}^{-})\right)^{2}\right]$$

By defining SpecLoss through spectral decomposition, its generalization error can be bounded using the recoverability and separability assumptions, which can also be understood in terms of the structure of the similarity graph.

Indeed, in Sec. 4.3, we demonstrated how GGAs and GED influence recoverability and separability by deriving an analogous generalization bound for SpecLoss that is tailored for graph data. At a high-level, to find this bound, we derived expressions for recoverability, α, and separability, ρ, based on graph edit distance, and then used these expression to recover the SpecLoss bound. We then performed some additional manipulation to derive the final expression presented in Thm. 6. Here, we provide the details and proofs behind these steps. We begin by restating the Separability plus Recoverability assumption.

Assumption 2 (**Separability plus Recoverability Assumption**, (Reproduction of Assm. 3.3))**.** *Let $\overline{\boldsymbol{g}} \in \overline{\mathcal{X}}$ and $y(\overline{\boldsymbol{g}})$ be its label, and $\boldsymbol{g} \sim \mathcal{A}(\cdot|\overline{\boldsymbol{g}})$. Assume that there exists a classifier h, such that $h(\boldsymbol{g}) = y(\overline{\boldsymbol{g}})$ with probability at least $1-\alpha$. We refer to α as the error of h.*

Now, recall from Sec. 4.3, that h will incur irreducible error on inconsistent samples, which are defined as follows:

Corollary 8. ***(Co-occuring augmentations.**,Reproduction of Coll. 3.4) Let $\overline{\boldsymbol{g}} \in \overline{\mathcal{X}}$ and $\boldsymbol{g}, \boldsymbol{g}' \in \mathcal{X}$. Then, $\boldsymbol{g} \sim \mathcal{A}(\overline{\boldsymbol{g}}) \wedge \boldsymbol{g}' \sim \mathcal{A}(\overline{\boldsymbol{g}}) \Leftrightarrow GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta$, where $\delta = \min\{\lfloor \gamma|\mathcal{V}_{\overline{\boldsymbol{g}}}|\rfloor, \lfloor \gamma|\mathcal{E}_{\overline{\boldsymbol{g}}}|\rfloor\ \lfloor \gamma|\mathcal{V}_{\boldsymbol{g}}|\rfloor, \lfloor \gamma|\mathcal{E}_{\boldsymbol{g}}|\rfloor\}$.*

Proof. Recall, that $\boldsymbol{g} \sim \mathcal{A}(\overline{\boldsymbol{g}}) \iff GED(\boldsymbol{g}, \overline{\boldsymbol{g}}) \leq \delta$ and $\boldsymbol{g}' \sim \mathcal{A}(\overline{\boldsymbol{g}}) \iff GED(\boldsymbol{g}', \overline{\boldsymbol{g}}) \leq \delta$. Then, $GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta$ and are co-occurring augmentations as they both belong to $\mathcal{A}(\overline{\boldsymbol{g}})$. $\square$

Definition 9 (**Inconsistent Samples**, Reproduction of Defn. 3.5)**.** *Let $\boldsymbol{g} \in \mathcal{X}$, and $y : \overline{\mathcal{X}} \to r$ be a labeling function. Further, let $\overline{\mathcal{X}}_{in} = \{\overline{\boldsymbol{g}} | \overline{\boldsymbol{g}} \in \overline{\mathcal{X}} \wedge GED(\boldsymbol{g}, \overline{\boldsymbol{g}}) \leq \delta\}$ be the set of natural samples that may have generated $\boldsymbol{g}$ and $Y^*_{in} = \{y(\overline{\boldsymbol{g}}) | \overline{\boldsymbol{g}} \in \overline{\mathcal{X}}_{in}\}$ be the set of unique labels. If $\boldsymbol{g}$ is an inconsistent sample, $|Y^*_{in}| > 1$.*

Now, we fix the behavior of h on inconsistent samples such that $h(\boldsymbol{g}) = y$, for some fixed $y \in Y^*_{in}$. Then, h induces an r-way partition over $\mathcal{X}$, such that each sample, $\boldsymbol{g}$, belongs to a partition, $\mathbf{S}_h(\boldsymbol{g})$. Further, because h will always incur error on inconsistent samples, α can be lower bounded by the ratio of inconsistent to total samples. To this end, we use GED to identify inconsistent samples by identifying disagreement amongst partitions as follows.

Lemma 5 (**Using GED to identify inconsistent samples**, Reproduction of Lemma 3.6)**.** *Let $\boldsymbol{g}, \boldsymbol{g}' \in \mathcal{X}$ and $GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta$ such that $\boldsymbol{g} \in \mathbf{S}_i \wedge \boldsymbol{g}' \in \mathbf{S}_j$ and $i \neq j$, where partitions are induced by h. Then, at least one $\tilde{\boldsymbol{g}} \in \{\boldsymbol{g}, \boldsymbol{g}'\}$ must be an inconsistent sample.*

Proof. By definition, $GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta$ implies that at least one of the following must be true: (i) $\overline{\boldsymbol{g}}_1 \in \overline{\mathcal{X}} \ni y(\overline{\boldsymbol{g}}_1) = i \wedge GED(\overline{\boldsymbol{g}}_1, \boldsymbol{g}) \leq \delta \wedge GED(\overline{\boldsymbol{g}}_1, \boldsymbol{g}') \leq \delta$ or (ii) $\overline{\boldsymbol{g}}_2 \in \overline{\mathcal{X}} \ni y(\overline{\boldsymbol{g}}_2) = j \wedge GED(\overline{\boldsymbol{g}}_2, \boldsymbol{g}) \leq \delta \wedge GED(\overline{\boldsymbol{g}}_2, \boldsymbol{g}') \leq \delta$. WLOG, assume (i). Now, $\boldsymbol{g}' \in \mathbf{S}_j \Leftrightarrow h(\boldsymbol{g}) = j$, so $j \in |Y^*_{in}|$. However, $GED(\bar{\boldsymbol{g}}_1, \boldsymbol{g}) \leq \delta$, so by Lemma 1 and Defn. 3, $y(\bar{\boldsymbol{g}}_1) = i \in Y^*_{in}$. Since, $i \neq j$, $|Y^*_{in}| > 1$, $\boldsymbol{g}$ must be an inconsistent sample. Note, if (ii) holds, then $\boldsymbol{g'}$ is an inconsistent sample. □

Note that the above lemma does not rely on ground-truth label information to identify inconsistent samples, but only GED from natural samples. Given that the error on inconsistent samples is irreducible, as it is unclear which $y \in Y_{in}$ is correct, we can lower bound the error of h as follows:

Corollary 10 (**Error bound due to Inconsistent Samples**, Reproduction of Coll. 3.7)**.** *The error of h can be lower-bounded as*

$$\alpha \geq \frac{\sum_i^r \sum_{\boldsymbol{g} \in S_i, \boldsymbol{g'} \notin S_i} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g'}) \leq 2\delta)}{|\mathcal{X}|}.$$

Here, the number of inconsistent samples can be approximated via $\sum_i^r \sum_{\boldsymbol{g} \in S_i, \boldsymbol{g'} \notin S_i} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g'}) \leq 2\delta)$ and $|\mathcal{X}|$ can be estimated using a combinatorial counting procedure. Thus, the above corollary reflects the fact that error on inconsistent samples cannot be reduced due to label un-identifiability.

Partition dissimilarity, which induces a notion of clustering of similar data-points in our analysis, can be defined as the following:

Definition 11 (**Partition Dissimilarity**, Reproduction of Defn. 3.8)**.** *Let $S_1, \dots, S_r$ be an r-way partition of $\mathcal{X}$. Then, we define the partition dissimilarity for a given partition as*

$$\phi_{\mathcal{X}}(S_i) = \frac{\sum_{\boldsymbol{g} \in S, \boldsymbol{g'} \notin S} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g'}) \leq 2\delta)}{\sum_{\boldsymbol{g} \in S} |\{\boldsymbol{g'} | GED(\boldsymbol{g}, \boldsymbol{g'}) \leq 2\delta\}|}.$$

We can now state the main result that re-derives the generalization error of SpecLoss in terms of GGAs, using the definitions of co-occurring pairs (Def. 2) and dissimilar partitions (Def. 5). Notably, we decompose bound in terms of the number of co-occurring augmentation-pairs <u>within</u> the same partition and the number of pairs that cross partitions, which are defined respectively as, $\lambda = \sum_{\boldsymbol{g} \in S_*, \boldsymbol{g'} \in S_*} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g'}) \leq 2\delta)$, and $\mu = \sum_{\boldsymbol{g} \in S_*, \boldsymbol{g'} \notin S_*} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g'}) \leq 2\delta)$.

Theorem 12 (**Generalization Bound for SpecLoss with GGA**, Reproduction of Thm 3.9)**.** *Assume the representation dimension $k \geq 2r$ and Assm. 4 holds for $\alpha \geq 0$. Let F be a hypothesis class containing a minimizer f^*_{pop} of SpecLoss, $\mathcal{L}(f)$, which produces a $\lfloor k/2 \rfloor$-way partition of $\mathcal{X}$ denoted by $\{S_*\}$. Let its most dissimilar partition have dissimilarity denoted by $\rho_{\lfloor k/2 \rfloor} = \min_i \phi(S_i \in \{S_*\})$. Then, f^*_{pop} has a generalization error bounded as, where the* *middle term is from the original SpecLoss bound:*

$$\mathcal{E}(f^*_{pop}) \leq \widetilde{O}\left(\alpha/\rho^2_{\lfloor k/2 \rfloor}\right) = \widetilde{O}\left(\frac{r}{|\mathcal{X}|}\left[\mu + 2\lambda + \frac{\lambda^2}{\mu}\right]\right),$$

Proof. The conversion from recoverability (α) and conductance (ρ) and within partition (μ) and across partition pairs (λ), can be derived as follows. We assume that the data distribution is I.I.D and the size of the class partitions are roughly equivalent.

$$\begin{aligned}\mathcal{E}(f^*_{pop}) &\leq \widetilde{O}\left(\alpha/\rho^2_{\lfloor k/2 \rfloor}\right)\\ &= \widetilde{O}\left(\frac{\sum_i^r \sum_{\boldsymbol{g}\in S_i, \boldsymbol{g}'\notin S_i} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)}{|\mathcal{X}|} \frac{1}{\left[\frac{\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)}{\sum_{x\in S_*} w_x}\right]^2}\right)\end{aligned}$$

$$\begin{aligned}
&\mathcal{E}(f^*_{pop}) \leq \widetilde{O}\left(\alpha/\rho^2_{\lfloor k/2 \rfloor}\right)\\
&= \widetilde{O}\left(\frac{\sum_i^r \sum_{\boldsymbol{g}\in S_i, \boldsymbol{g}'\notin S_i} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)}{|\mathcal{X}|} \frac{\left[\sum_{x\in S_*} w_x\right]^2}{\left[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)\right]^2}\right)\\
&= \widetilde{O}\left(\frac{r\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)}{|\mathcal{X}|} \frac{\left[\sum_{x\in S_*} w_x\right]^2}{\left[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)\right]^2}\right)\\
&= \widetilde{O}\left(\frac{r\left[\sum_{x\in S_*} w_x\right]^2}{|\mathcal{X}|\left[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)\right]}\right)\\
&= \widetilde{O}\left(\frac{r\left[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta) + \sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\in S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)\right]^2}{|\mathcal{X}|\left[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)\right]}\right)\\
&= \widetilde{O}\Bigg(\frac{r}{|\mathcal{X}|}\Bigg[\frac{\left[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)\right]^2}{\left[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)\right]}\\
&+ \frac{2\left[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\in S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)\right]}{\left[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)\right]}\\
&+ \frac{\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\in S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)}{\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)}\Bigg]\Bigg)\\
&= \widetilde{O}\Bigg(\frac{r}{|\mathcal{X}|}\Bigg[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)\\
&+ 2\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\in S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta) + \frac{\left[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\in S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)\right]^2}{\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g},\boldsymbol{g}') \leq 2\delta)}\Bigg]\Bigg)
\end{aligned} \tag{B.1}$$

Now, notice that the above equation can be understood as the number of inconsistent samples vs. the original samples. Let, $\lambda = \sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\in S_*} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta)$ and $\mu = \sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta)$. Then, we have recovered the bound presented in Theorem 6.

$$\begin{aligned}\widetilde{O}\left(\alpha/\rho^2_{\lfloor k/2 \rfloor}\right) &= \widetilde{O}\Bigg(\frac{r}{|\mathcal{X}|}\Bigg[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta) \\ &\quad + 2\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\in S_*} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta) + \frac{\left[\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\in S_*} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta)\right]^2}{\sum_{\boldsymbol{g}\in S_*, \boldsymbol{g}'\notin S_*} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta)}\Bigg]\Bigg) \\ &\approx \widetilde{O}\left(\frac{r}{|\mathcal{X}|}\left[\underbrace{\mu}_{\text{inconsistent samples}} + \underbrace{2\lambda}_{\text{valid samples}} + \frac{\overbrace{\lambda^2}^{\text{valid samples}}}{\underbrace{\mu}_{\text{inconsistent samples}}}\right]\right).\end{aligned} \tag{B.2}$$

Recall, that inconsistent samples can be determined through graph edit distance (Defn. 3) between augmented samples. Moreover, that the maximum allowable edit distance between augmented samples is determined by augmentation strength. □

B.5.2 Connections to the Population Augmentation Graph

The original bound for SpecLoss uses the population augmentation graph (PAG). While we did not use the PAG in our analysis for ease of exposition, we note that our analysis can be adapted for the PAG as follows:

Definition 13 (Population Augmentation Graph [**?**]). *Let $\mathcal{G}^p$ be the PAG where the vertex set is all augmented data $\mathcal{X}$. For any two augmented data $\boldsymbol{g}, \boldsymbol{g}' \in \mathcal{X}$, define the edge weight $w_{\boldsymbol{g}\boldsymbol{g}'}$ as the marginal probability of generating $\boldsymbol{g}$ and $\boldsymbol{g}'$ from a random natural data $\overline{\boldsymbol{g}} \sim \mathcal{P}_{\overline{\mathcal{X}}}$:*

$$w_{\boldsymbol{g}\boldsymbol{g}'} := \mathbb{E}_{\overline{\boldsymbol{g}}\in\mathcal{P}_{\overline{\mathcal{X}}}}[\mathcal{A}(\boldsymbol{g}|\overline{\boldsymbol{g}})\mathcal{A}(\boldsymbol{g}'|\overline{\boldsymbol{g}})]. \tag{B.3}$$

To extend our analysis to the PAG, we show that connectivity in the PAG is also determined by GED. Then, the definition of inconsistent samples, and partition dissimilarity (conductance) straight-forwardly follow.

Lemma 6. ***Connectivity in the PAG is determined by GED.*** *Let $\boldsymbol{g}, \boldsymbol{g}' \in \mathcal{X}$, and $\overline{\boldsymbol{g}} \in \overline{\mathcal{X}}$. Then, $w_{\boldsymbol{g}\boldsymbol{g}'} > 0 \Leftrightarrow GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta$.*

Proof. By Lemma 2, $w_{\boldsymbol{g}\boldsymbol{g}'} > 0 \Leftrightarrow \mathcal{A}(\boldsymbol{g}|\overline{\boldsymbol{g}}) > 0 \wedge \mathcal{A}(\boldsymbol{g}'|\overline{\boldsymbol{g}}) > 0$. Moreover, if $\mathcal{A}(\boldsymbol{g}|\overline{\boldsymbol{g}}) > 0$ then, $\boldsymbol{g}$ is the augmentation set of $\overline{\boldsymbol{g}}$. If $\boldsymbol{g} \in \mathcal{A}(\overline{\boldsymbol{g}})$ then, $GED(\boldsymbol{g}, \overline{\boldsymbol{g}}) \leq \delta$. Then, $w_{\boldsymbol{g}\boldsymbol{g}'} > 0 \Leftrightarrow GED(\boldsymbol{g}, \overline{\boldsymbol{g}}) \leq \delta \wedge GED(\boldsymbol{g}', \overline{\boldsymbol{g}}) \leq \delta$, which in turn applies, $w_{\boldsymbol{g}\boldsymbol{g}'} > 0 \Leftrightarrow GED(\boldsymbol{g}, \boldsymbol{g}') \leq 2\delta$. □

Corollary 14 (Conductance according to GGA). *Recall, the conductance ϕ_G of a partition S_i in a graph G measures how many edges cross partitions relative to total number of edges a node possesses and that $\mathcal{A}(\boldsymbol{g}|\overline{\boldsymbol{g}}) \approx \frac{1}{|\mathcal{A}(\overline{\boldsymbol{g}})|}$. Then,*

$$\phi_G(S_i) = \frac{\sum_{x \in S, x' \notin S} \mathbb{1}(w_{xx'} > 0)}{\sum_{x \in S} w_x},$$

where w_x represents the size of x's edge-set.

Using this definition, we can substitute into the original SpecLoss generalization bound and recover the result presented in Thm. 6.

B.6 Dataset Generation and Experimental Details

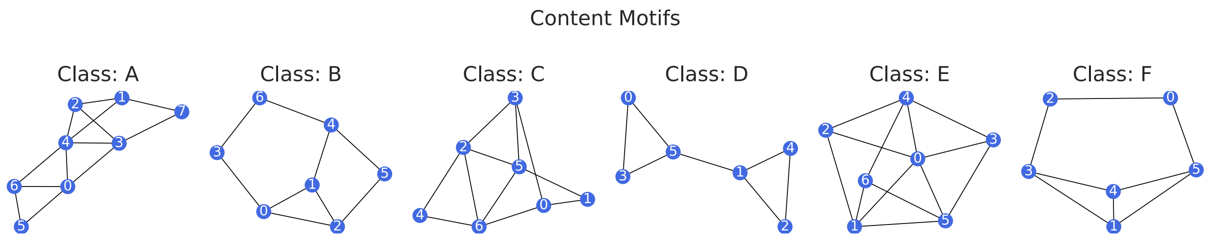


Figure B.2: **Motifs used to determine class labels.**

We use the motifs shown in Fig. B.6 to define a 6 class graph classification task. It is important to ensure that the motifs are not isomorphic, as many GNNs are less expressive than the 1-Weisfeiler Lehman's test for isomorphism ([27]). For each class, 1000 random samples are generated as follows: (i) We randomly select between 1-3 motifs to be in each sample. At this time, motifs all belong to the same class, though this condition could easily be changed for a more difficult task. (ii) We define the number of content nodes, C_n, as the size of the selected motif, scaled by the number of motifs in the sample. (iii) For a given style ratio, we determine the number of possible style nodes as $S_n = \rho C_n$ (iv). We define $RBG(n)$ using networkx's * random tree generator: `networkx.generators.trees.random_tree`. We note that other random graph generators would also be well suited for this task. (v) For additional randomness, we create background graphs using $S_n \pm 2$, and also randomly perturb up-to 10% of edges in sample. We repeat this set-up with $\rho \in \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.5, 8.0\}$ to generate the datasets used in Sec 4.4.2.

*https://networkx.org/documentation/stable/

Table B.3: **Dataset Description**

Name	**Graphs**	**Classes**	**Avg. Nodes**	**Avg. Edges**	**Domain**
IMDB-BINARY [136]	1000	2	19.77	96.53	Social
REDDIT-BINARY [136]	2000	2	429.63	497.75	Social
MUTAG [138]	188	2	17.93	19.79	Molecule
PROTEINS [139]	1113	2	39.06	72.82	Bioinf.
DD [140]	1178	2	284.32	715.66	Bioinf.
NCI1 [93]	4110	2	29.87	32.30	Molecule

Experimental Set-up: We follow You et al. [1] for TUDataset experiments. When reporting the kNN accuracy, we tune $k \in \{5, 10, 15, 20\}$ separately on validation data for each dataset and method to allow for the strongest baselines. For synthetic datasets we use the following setup. Our encoder is a 5-layer GIN model with mean pooling. We set input node features to be a constant 10-dimensional feature vector, and a hidden layer dimension is 32; we concatenate hidden representations for a representation dimension of 160. Models are pretrained for 60 epochs. Subsequently, we use a linear evaluation protocol and train a linear head for 200 epochs. All models are trained with Adam, lr = 0.01.

B.7 Related Work

Table B.4: **Selected Graph Contrastive Learning Frameworks.** We provide a brief description of augmentations used by selected frameworks. Most frameworks use random corruptive, sampling, or diffusion-based approaches to generate augmentations.

Method	Augmentations
GraphCL ([1])	Node Dropping, Edge Adding/Dropping, Attribute Masking, Subgraph Extraction
GCC ([272])	RWR Subgraph Extraction of Ego Network
MVGRL ([51])	PPR Diffusion + Sampling
GCA ([149])	Edge Dropping, Attribute Masking (both weighted by centrality)
BGRL ([52])	Edge Dropping, Attribute Masking
SelfGNN ([131])	Attribute Splitting, Attribute Standardization + Scaling, Local Degree Profile, Paste + Local Degree Profile

Graph Data Augmentation: Unlike images, graphs are discrete objects that do not naturally lie in Euclidiean space, making it difficult to define meaningful augmentations. Furthermore, while for images or natural language, there may be an intuitive understanding of what changes will preserve task-relevant information, this is not the case for graphs. Indeed, a single edge change can completely change the properties of a molecular graph. Therefore, only a few works consider graph data augmentation. [113] note that a node classification task can be perfectly solved if edges only exist between same class samples. They increase homophily by adding edges between nodes that a neural network predicts belong to the same class and breaking edges between nodes of predicted dissimilar classes. However, this approach is expensive and not applicable to graph classification. [112] argue that information preserving topological transformations are difficult for the aforementioned reasons and instead focus on feature augmentations. Throughout training, they add an adversarial perturbation to node features to improve generalization, computing the gradient of the model weights while computing the gradients of the adversarial perturbation to avoid more expensive adversarial training [277]. This approach is not directly applicable to contrastive learning, where label information cannot be used to generate the adversarial perturbation.

Graph Self-Supervised Learning: In graphs, recent works have explored several paradigms for self-supervised learning: see [273] for an up-to-date survey. Graph pre-text tasks are often reminiscent of image in-painting tasks [274], and seek to complete masked graphs and/or node features ([275, 95]). Other successful approaches include predicting auxiliary properties of nodes or entire graphs during pre-training or part of regular training to prevent overfitting ([95]). These tasks often must be carefully selected to avoid negative transfer between tasks. Many contrast-based unsupervised approaches have also been proposed, often inspired by techniques designed for non-graph data. [14, 276] draw inspiration from [123] and maximize the mutual information between global and local representations. MVGRL ([51]) contrasts different views at multiple granularities similar to [114]. [1, 272, 149, 52, 131] use augmentations (which we summarize in Table B.7) to generate views for contrastive learning. We note that random corruption, sampling or diffusion based approaches used to create generic graph augmentations often do not preserve task-relevant information or introduce meaningful invariances.

APPENDIX C

Uncertainty Quantification with GNNs

C.1 Ethics Statement

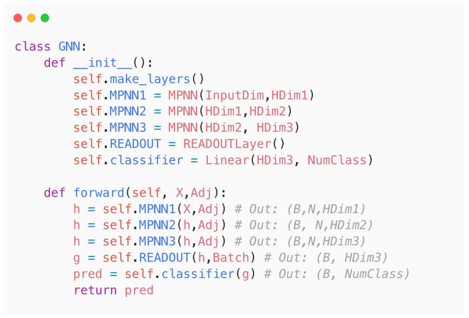
This work proposes a method to improve uncertainty estimation in graph neural networks, which has potential broader societal impacts. As graph learning models are increasingly deployed in real-world applications like healthcare, finance, and transportation, it becomes crucial to ensure these models make reliable predictions and know when they may be wrong. Unreliable models can lead to harmful outcomes if deployed carelessly. By improving uncertainty quantification, our work contributes towards trustworthy graph AI systems.

We also consider several additional safety-critical tasks, including generalization gap prediction for graph classification (to the best of our knowledge, we are the first to report results on this task) and OOD detection. We hope our work will encourage further study in these important areas.

However, there are some limitations. Our method requires (modest) additional computation during training and inference, which increases resource usage. Although G-ΔUQ, unlike post-hoc methods, does not need to be fit on a validation dataset, evaluation of its benefits also also relies on having some out-of-distribution or shifted data available, which may not always be feasible. We have seen in Table 5.1 that there are tasks for which G-ΔUQfails to improve accuracy and/or calibration of some post-hoc methods, further emphasizing the need to perform appropriate model selection and the risks if shifted validation data is not available. Finally, there are open questions around how much enhancement in uncertainty calibration translates to real-world safety and performance gains.

Looking ahead, we believe improving uncertainty estimates is an important direction for graph neural networks and deep learning more broadly. This will enable the development safe, reliable AI that benefits society. We hope our work inspires more research in the graph domain that focuses on uncertainty quantification and techniques that provide guarantees about model behavior, especially for safety-critical applications. Continued progress will require interdisciplinary collaboration between graph machine learning researchers and domain experts in areas where models are deployed.

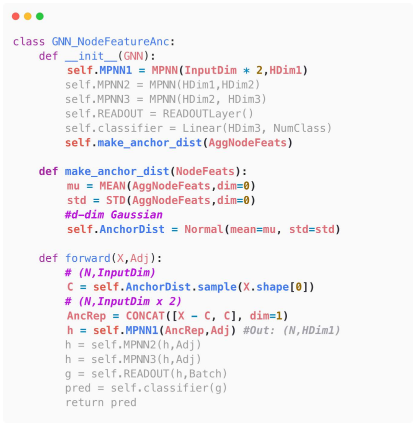
C.2 PseudoCode



```
class GNN:
    def __init__():
        self.make_layers()
        self.MPNN1 = MPNN(InputDim,HDim1)
        self.MPNN2 = MPNN(HDim1,HDim2)
        self.MPNN3 = MPNN(HDim2, HDim3)
        self.READOUT = READOUTLayer()
        self.classifier = Linear(HDim3, NumClass)

    def forward(self, X,Adj):
        h = self.MPNN1(X,Adj) # Out: (B,N,HDim1)
        h = self.MPNN2(h,Adj) # Out: (B, N,HDim2)
        h = self.MPNN3(h,Adj) # Out: (B,N,HDim3)
        g = self.READOUT(h,Batch) # Out: (B, HDim3)
        pred = self.classifier(g) # Out: (B, NumClass)
        return pred
```

(a) Vanilla GNN

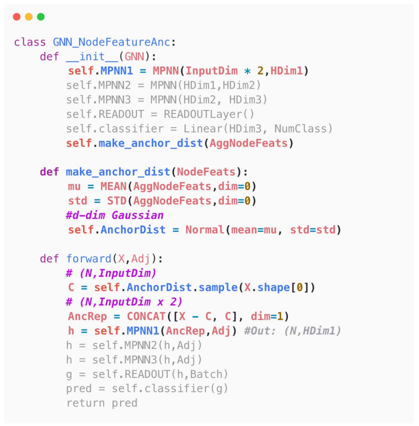


```
class GNN_NodeFeatureAnc:
    def __init__(GNN):
        self.MPNN1 = MPNN(InputDim * 2,HDim1)
        self.MPNN2 = MPNN(HDim1,HDim2)
        self.MPNN3 = MPNN(HDim2, HDim3)
        self.READOUT = READOUTLayer()
        self.classifier = Linear(HDim3, NumClass)
        self.make_anchor_dist(AggNodeFeats)

    def make_anchor_dist(NodeFeats):
        mu = MEAN(AggNodeFeats,dim=0)
        std = STD(AggNodeFeats,dim=0)
        #d-dim Gaussian
        self.AnchorDist = Normal(mean=mu, std=std)

    def forward(X,Adj):
        # (N,InputDim)
        C = self.AnchorDist.sample(X.shape[0])
        # (N,InputDim x 2)
        AncRep = CONCAT([X - C, C], dim=1)
        h = self.MPNN1(AncRep,Adj) #Out: (N,HDim1)
        h = self.MPNN2(h,Adj)
        h = self.MPNN3(h,Adj)
        g = self.READOUT(h,Batch)
        pred = self.classifier(g)
        return pred
```

(b) G-ΔUQ with Node Feature Anchoring

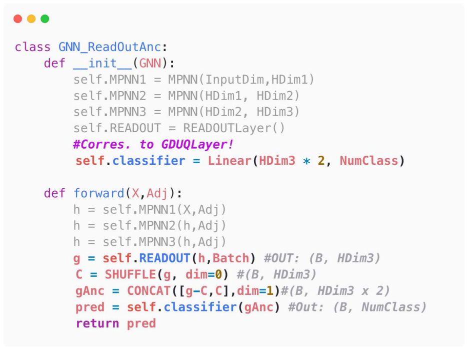


```
class GNN_NodeFeatureAnc:
    def __init__(GNN):
        self.MPNN1 = MPNN(InputDim * 2,HDim1)
        self.MPNN2 = MPNN(HDim1,HDim2)
        self.MPNN3 = MPNN(HDim2, HDim3)
        self.READOUT = READOUTLayer()
        self.classifier = Linear(HDim3, NumClass)
        self.make_anchor_dist(AggNodeFeats)

    def make_anchor_dist(NodeFeats):
        mu = MEAN(AggNodeFeats,dim=0)
        std = STD(AggNodeFeats,dim=0)
        #d-dim Gaussian
        self.AnchorDist = Normal(mean=mu, std=std)

    def forward(X,Adj):
        # (N,InputDim)
        C = self.AnchorDist.sample(X.shape[0])
        # (N,InputDim x 2)
        AncRep = CONCAT([X - C, C], dim=1)
        h = self.MPNN1(AncRep,Adj) #Out: (N,HDim1)
        h = self.MPNN2(h,Adj)
        h = self.MPNN3(h,Adj)
        g = self.READOUT(h,Batch)
        pred = self.classifier(g)
        return pred
```

(c) G-ΔUQ with Hidden Rep Anchoring



```
class GNN_ReadOutAnc:
    def __init__(GNN):
        self.MPNN1 = MPNN(InputDim,HDim1)
        self.MPNN2 = MPNN(HDim1, HDim2)
        self.MPNN3 = MPNN(HDim2, HDim3)
        self.READOUT = READOUTLayer()
        #Corres. to GDUQLayer!
        self.classifier = Linear(HDim3 * 2, NumClass)

    def forward(X,Adj):
        h = self.MPNN1(X,Adj)
        h = self.MPNN2(h,Adj)
        h = self.MPNN3(h,Adj)
        g = self.READOUT(h,Batch) #OUT: (B, HDim3)
        C = SHUFFLE(g, dim=0) #(B, HDim3)
        gAnc = CONCAT([g-C,C],dim=1)#(B, HDim3 x 2)
        pred = self.classifier(gAnc) #Out: (B, NumClass)
        return pred
```

(d) G-ΔUQ with READOUT Anchoring

Figure C.1: **PseudoCode for G-ΔUQ**. We provide simplified pseudo-code to demonstrate how anchoring can be performed. We assume PyTorchGeometric style mini-batching. Changes with respect to the vanilla GNN are shown in bold. Unchanged lines are grayed out.

C.3 Reproducibility

For reproducing our experiments, we have made our code available at this repository. In the remainder of this appendix (specifically App. C.6, C.7), and C.10), we also provide additional details about the benchmarks and experimental setup.

C.4 Details on Super-pixel Experiments

We provide an example of the rotated images and corresponding super-pixel graphs in Fig. C.2. (Note that classes “6” and “9” may be confused under severe distribution shift, i.e. 90 degrees rotation or more. Hence, to avoid harming class information, our experiments only consider distribution shift from rotation up to 40 degrees.)

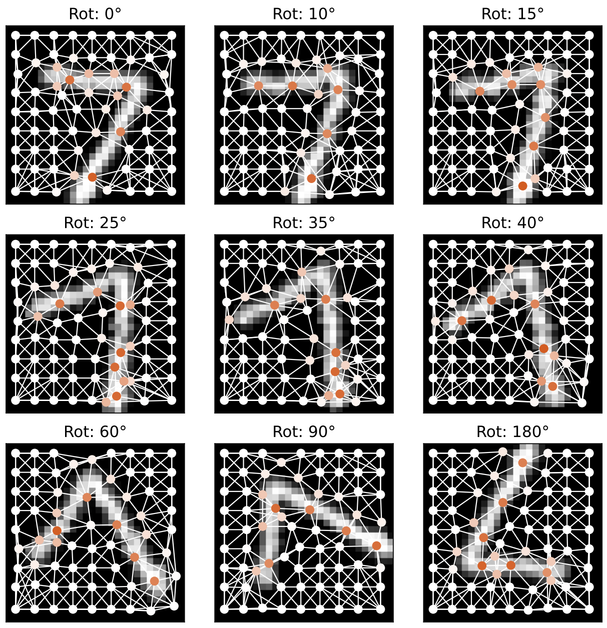


Figure C.2: **Rotated Super-pixel MNIST.** Rotating images prior to creating super-pixels to leads to some structural distortion [12]. However, we can see that the class-discriminative information is preserved, despite rotation. This allows for simulating different levels of graph structure distribution shifts, while still ensuring that samples are valid.

Tables C.1 and C.2 provided expanded results on the rotated image super-pixel graph classification task, discussed in Sec. 5.6.1.

In Table C.4 we focus on the calibration results on this task for GPS variants alone. Across all levels of distribution shift, the best method is our strategy for applying G-ΔUQ to a pretrained model–demonstrating that this is not just a practical choice when it is infeasible to retrain a model, but can lead to powerful performance by any measure. Second-best on all datasets is applying G-ΔUQ during training, further highlighting the benefits of stochastic anchoring.

In addition to the structural distribution shifts we get by rotating the images before constructing super-pixel graphs, we also simulate feature distribution shifts by adding

Table C.1: **RotMNIST-Accuracy.** Here, we report expanded results (accuracy) on the Rotated MNIST dataset, including a variant that combines G-ΔUQ with Deep Ens. Notably, we see that anchored ensembles outperform basic ensembles in both accuracy and calibration. (Best results for models using Deep Ens. and those not using it marked separately.)

MODEL	G-ΔUQ?	LPE?	Avg. Test (↑)	Acc. (10) (↑)	Acc. (15) (↑)	Acc. (25) (↑)	Acc. (35) (↑)	Acc. (40) (↑)
GatedGCN	×	×	0.947 ±0.002	0.918 ±0.002	0.904 ±0.005	0.828 ±0.009	0.738 ±0.009	0.679 ±0.007
	✓	×	0.933 ±0.015	0.894 ±0.019	0.878 ±0.020	0.794 ±0.032	0.698 ±0.036	0.636 ±0.048
	×	✓	0.949 ±0.002	0.917 ±0.004	0.904 ±0.005	0.829 ±0.007	0.744 ±0.007	0.685 ±0.006
	✓	✓	0.915 ±0.032	0.872 ±0.038	0.852 ±0.0414	0.776 ±0.039	0.680 ±0.037	0.631 ±0.033
GPS	×	✓	**0.970** ±0.001	**0.948** ±0.001	**0.938** ±0.001	**0.873** ±0.006	**0.770** ±0.013	**0.688** ±0.009
	✓	✓	0.969 ±0.001	0.946 ±0.003	0.937 ±0.003	0.869 ±0.003	0.769 ±0.012	0.679 ±0.014
GPS (Pretrained)	✓	✓	0.967 ±0.002	0.945 ±0.004	0.934 ±0.005	0.864 ±0.009	0.759 ±0.010	0.674 ±0.002
GatedGCN-DENS	×	×	0.963 ±0.0002	0.943 ±0.001	0.933 ±0.001	0.874 ±0.002	0.794 ±0.002	0.731 ±0.002
	✓	×	0.949 ±0.008	0.922 ±0.008	0.907 ±0.011	0.828 ±0.020	0.733 ±0.032	0.662 ±0.046
	×	✓	0.965 ±0.001	0.943 ±0.001	0.933 ±0.001	0.873 ±0.001	0.792 ±0.004	0.736 ±0.003
	✓	✓	0.954 ±0.005	0.930 ±0.010	0.917 ±0.011	0.850 ±0.023	0.759 ±0.025	0.696 ±0.032
GPS-DENS	×	✓	**0.980** ±0.000	**0.969** ±0.000	**0.961** ±0.000	**0.913** ±0.000	**0.834** ±0.000	**0.750** ±0.000
	✓	✓	0.978 ±0.001	0.963 ±0.000	0.953 ±0.001	0.905 ±0.000	0.822 ±0.002	0.736 ±0.003

Gaussian noise with different standard deviations to the pixel value node features in the super-pixel graphs. In Table C.5, we report accuracy and calibration results for varying levels of distribution shift (represented by the size of the standard deviation of the Gaussian noise). Across different levels of feature distribution shift, we also see that G-ΔUQ results in superior calibration, while maintaining competitive or in many cases superior accuracy.

Table C.2: **RotMNIST-Calibration.** Here, we report expanded results (calibration) on the Rotated MNIST dataset, including a variant that combines G-ΔUQ with Deep Ens. Notably, we see that anchored ensembles outperform basic ensembles in both accuracy and calibration. (Best results for models using Deep Ens. and those not using it marked separately.)

MODEL	G-ΔUQ	LPE?	Avg.ECE (↓)	ECE (10) (↓)	ECE (15) (↓)	ECE (25) (↓)	ECE (35) (↓)	ECE (40) (↓)
GatedGCN-TS	×	×	0.035 ±0.001	0.054 ±0.002	0.062 ±0.003	0.118 ±0.007	0.185 ±0.006	0.233 ±0.008
	×	✓	0.033 ±0.002	0.053 ±0.002	0.061 ±0.004	0.116 ±0.005	0.179 ±0.006	0.225 ±0.005
GatedGCN	×	×	0.038 ±0.001	0.059 ±0.001	0.068 ±0.340	0.126 ±0.008	0.195 ±0.012	0.245 ±0.011
	✓	×	**0.018** ±0.008	0.029 ±0.013	**0.033** ±0.164	0.069 ±0.033	0.117 ±0.048	0.162 ±0.067
	×	✓	0.036 ±0.003	0.059 ±0.002	0.068 ±0.340	0.125 ±0.006	0.191 ±0.007	0.240 ±0.008
	✓	✓	0.022 ±0.007	**0.028** ±0.014	0.034 ±0.169	**0.062** ±0.022	**0.109** ±0.019	**0.141** ±0.019
GPS-TS	×	✓	0.024 ±0.001	0.041 ±0.001	0.049 ±0.001	0.102 ±0.006	0.188 ±0.012	0.261 ±0.008
GPS	×	✓	0.026 ±0.001	0.044 ±0.001	0.052 ±0.156	0.108 ±0.006	0.197 ±0.012	0.273 ±0.008
	✓	✓	0.022 ±0.001	0.037 ±0.005	0.044 ±0.133	0.091 ±0.008	0.165 ±0.018	0.239 ±0.018
GPS (Pretrained)	✓	✓	0.021 ±0.001	0.032 ±0.003	0.039 ±0.116	0.083 ±0.002	0.153 ±0.007	0.217 ±0.012
GatedGCN-DENS	×	×	0.026 ±0.000	0.038 ±0.001	0.042 ±0.001	0.084 ±0.002	0.135 ±0.001	0.185 ±0.003
	✓	×	**0.014** ±0.003	**0.018** ±0.005	**0.021** ±0.005	0.036 ±0.012	0.069 ±0.032	0.114 ±0.056
	×	✓	0.024 ±0.001	0.038 ±0.001	0.043 ±0.002	0.083 ±0.001	0.139 ±0.004	0.181 ±0.002
	✓	✓	0.017 ±0.002	0.024 ±0.005	0.027 ±0.008	**0.030** ±0.004	**0.036** ±0.012	**0.059** ±0.033
GPS-DENS	×	✓	0.016 ±0.001	0.026 ±0.002	0.030 ±0.000	0.066 ±0.000	0.123 ±0.000	0.195 ±0.000
	✓	✓	**0.014** ±0.000	0.023 ±0.002	0.027 ±0.003	0.055 ±0.004	0.103 ±0.006	0.164 ±0.006

Table C.3: **Accuracy of GPS Variants on RotatedMNIST.** We focus on the accuracy results for GPS variants on rotated MNIST dataset. Using G-ΔUQ (with or without pretraining) remains close in accuracy to foregoing it, generally within the range of the standard deviation of the results.

MODEL	G-ΔUQ?	Avg. Test (↑)	Acc. (10) (↑)	Acc. (15) (↑)	Acc. (25) (↑)	Acc. (35) (↑)	Acc. (40) (↑)
GPS	×	**0.970** ±0.001	**0.948** ±0.001	**0.938** ±0.001	**0.873** ±0.006	**0.770** ±0.013	**0.688** ±0.009
	✓	0.969 ±0.001	0.946 ±0.003	0.937 ±0.003	0.869 ±0.003	0.769 ±0.012	0.679 ±0.014
GPS (Pretrained)	✓	0.967 ±0.002	0.945 ±0.004	0.934 ±0.005	0.864 ±0.009	0.759 ±0.010	0.674 ±0.002

Table C.4: **Calibration of GPS Variants on RotatedMNIST.** We focus on the calibration results for GPS variants on rotated MNIST dataset. Across the board, we see improvements from using G-ΔUQ , with our strategy of applying it to a pretrained model doing best.

MODEL	G-ΔUQ	Avg.ECE (↓)	ECE (10) (↓)	ECE (15) (↓)	ECE (25) (↓)	ECE (35) (↓)	ECE (40) (↓)
GPS-TS	×	0.024 ±0.001	0.041 ±0.001	0.049 ±0.001	0.102 ±0.006	0.188 ±0.012	0.261 ±0.008
GPS	×	0.026 ±0.001	0.044 ±0.001	0.052 ±0.156	0.108 ±0.006	0.197 ±0.012	0.273 ±0.008
	✓	0.022 ±0.001	0.037 ±0.005	0.044 ±0.133	0.091 ±0.008	0.165 ±0.018	0.239 ±0.018
GPS (Pretrained)	✓	**0.021** ±0.001	**0.032** ±0.003	**0.039** ±0.116	**0.083** ±0.002	**0.153** ±0.007	**0.217** ±0.012

Table C.5: **MNIST Feature Shifts**. G-ΔUQ improves calibration and maintains competitive or even improved accuracy across varying levels of feature distribution shift.

				STD = 0.1		STD = 0.2		STD = 0.3		STD = 0.4	
MODEL	LPE?	G-ΔUQ?	Calibration	Accuracy (↑)	ECE (↓)	Accuracy (↑)	ECE (↓)	Accuracy (↑)	ECE (↓)	Accuracy (↑)	ECE (↓)
GatedGCN	×	×	×	0.742±0.005	0.186±0.018	0.481±0.015	0.414±0.092	0.293±0.074	0.606±0.147	0.197±0.092	0.71±0.178
	×	✓	×	**0.773**±0.053	**0.075**±0.032	0.536±0.010	**0.160**±0.087	**0.356**±0.101	0.422±0.083	**0.249**±0.074	**0.529**±0.047
	✓	×	×	0.751±0.02	0.176±0.014	0.519±0.004	0.348±0.03	0.345±0.032	0.485±0.096	0.233±0.043	0.581±0.142
	✓	✓	×	0.745±0.026	0.100±0.036	**0.541**±0.040	0.235±0.067	**0.355**±0.062	**0.408**±0.116	0.242±0.063	0.539±0.139

C.5 Stochastic Centering on the Empirical NTK of GNNs

Using a simple grid-graph dataset and 4 layer GIN model, we compute the Fourier spectrum of the NTK. As shown in Fig. C.3, we find that shifts to the node features can induce systematic changes to the spectrum.

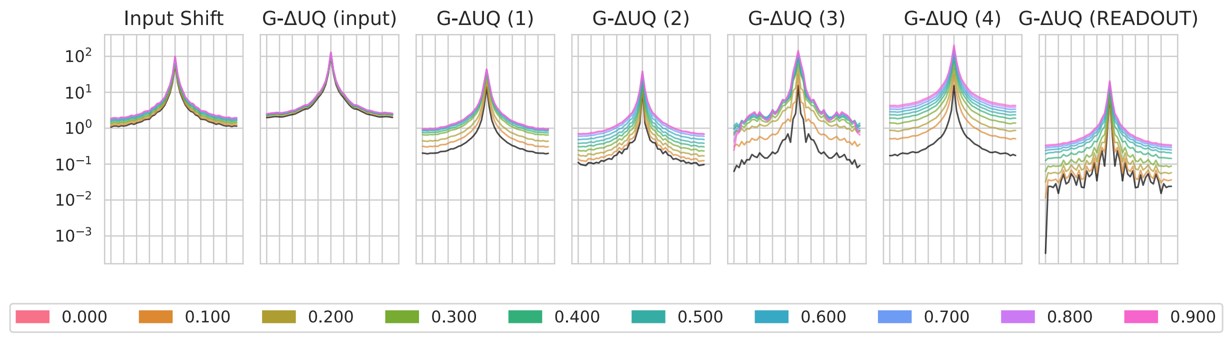


Figure C.3: **Stochastic Centering with the empirical GNN NTK.** We find that performing constant shifts at intermediate layers introduces changes to a GNN's NTK. We include a vanilla GNN NTK in black for reference. Further, note the shape of the spectrum should not be compared across subplots as each subplot was created with a different random initialization.

C.6 Size-Generalization Dataset Statistics

The statistics for the size generalization experiments (see Sec. 5.5.1) are provided below in Table C.6.

Table C.6: **Size Generalization Dataset Statistics:** This table is directly reproduced from [17], who in turn used statistics from [18, 19].

	NCI1			**NCI109**		
	all	**Smallest 50%**	**Largest 10%**	**all**	**Smallest 50%**	**Largest 10%**
Class A	49.95%	62.30%	19.17%	49.62%	62.04%	21.37%
Class B	50.04%	37.69%	80.82%	50.37%	37.95%	78.62%
# of graphs	4110	2157	412	4127	2079	421
Avg graph size	29	20	61	29	20	61

	PROTEINS			**DD**		
	all	**Smallest 50%**	**Largest 10%**	**all**	**Smallest 50%**	**Largest 10%**
Class A	59.56%	41.97%	90.17%	58.65%	35.47%	79.66%
Class B	40.43%	58.02%	9.82%	41.34%	64.52%	20.33%
# of graphs	1113	567	112	1178	592	118
Avg graph size	39	15	138	284	144	746

C.7 GOOD Benchmark Experimental Details

For our experiments in Sec. 5.5.2, we utilize the in/out-of-distribution covariate and concept splits provided by [35]. Furthermore, we use the suggested models and architectures provided by their package. In brief, we use GIN models with virtual nodes (except for GOODMotif) for training, and average scores over 3 seeds. When performing stochastic anchoring at a particular layer, we double the hidden representation size for that layer. Subsequent layers retain the original size of the vanilla model.

When performing stochastic anchoring, we use 10 fixed anchors randomly drawn from the in-distribution validation dataset. We also train the G-ΔUQ for an additional 50 epochs to ensure that models are able to converge. Please see our code repository for the full details.

We also include results on additional node classification benchmarks featuring distribution shift in Table C.9. In Table C.10, we compare models without G-ΔUQ to the use of G-ΔUQ with randomly sampled anchors at the first or second layer.

Dataset	Shift	Train	ID validation	ID test	OOD validation	OOD test	Train	OOD validation	ID validation	ID test	OOD test
				Length							
	covariate	24744	5301	5301	17206	17490					
GOOD-SST2	concept	27270	5843	5843	15142	15944					
				Color							
	covariate	42000	7000	7000	7000	7000					
GOOD-CMNIST	concept	29400	6300	6300	14000	14000					
	no shift	42000	14000	14000	-	-					
				Base					Size		
	covariate	18000	3000	3000	3000	3000	18000	3000	3000	3000	3000
GOOD-Motif	concept	12600	2700	2700	6000	6000	12600	2700	2700	6000	6000
				Word					Degree		
	covariate	9378	1979	1979	3003	3454	8213	1979	1979	3841	3781
GOOD-Cora	concept	7273	1558	1558	3807	5597	7281	1560	1560	3706	5686
				University							
	covariate	244	61	61	125	126					
GOOD-WebKB	concept	282	60	60	106	109					
				Color							
	covariate	420	70	70	70	70					
GOOD-CBAS	concept	140	140	140	140	140					

Table C.7: **Number of Graphs/Nodes per dataset.**

Dataset	Model	# Model layers	Batch Size	# Max Epochs	# Iterations per epoch	Initial LR	Node Feature Dim
GOOD-SST2	GIN-Virtual	3	32	200/100	–	1e-3	768
GOOD-CMNIST	GIN-Virtual	5	128	500	–	1e-3	3
GOOD-Motif	GIN	3	32	200	–	1e-3	4
GOOD-Cora	GCN	3	4096	100	10	1e-3	8710
GOOD-WebKB	GCN	3	4096	100	10	1e-3/5e-3	1703
GOOD-CBAS	GCN	3	1000	200	10	3e-3	8

Table C.8: **Model and hyperparameters for GOOD datasets.**

C.8 GOOD Dataset Additional Results

We also include results on additional node classification benchmarks featuring distribution shift in Table C.9. In Table C.10, we compare models without G-ΔUQ to the use of G-ΔUQ with randomly sampled anchors at the first or second layer.

Table C.9: **Additional Node Classification Benchmarks.** Here, we compare accuracy and calibration error of G-ΔUQ and "no G-ΔUQ " (vanilla) models on 4 node classification benchmarks across concept and covariate shifts. First, we note that across all our evaluations, without any posthoc calibration, G-ΔUQ is superior to the vanilla model on nearly every benchmark for better or same accuracy (8/8 benchmarks) and better calibration error (7/8), often with a significant gain in calibration performance. However, due to the challenging nature of these shifts, achieving state-of-the-art calibration performance often requires the use of post-hoc calibration methods – so we also evaluate how these posthoc methods can be elevated when combined with G-ΔUQ (versus the vanilla variant). When combined with popular posthoc methods, we highlight that performance improves across the board, when combined with G-ΔUQ (including in WebKB and CBAS-Concept). For example, on WebKB – across the 9 calibration methods considered, "G-ΔUQ + calibration method" improves or maintains the calibration performance of the analogous "no G-ΔUQ + calibration method" in 7/9 (concept) and 6/9 (covariate). In CBAS, calibration is improved or maintained as the no-G-ΔUQ version on 5/9 (concept) and 9/9 (covariate). In all cases, this is achieved with little or no compromise on classification accuracy (often improving over "no G-ΔUQ" variant). We also emphasize that, across all the 8 evaluation sets (4 datasets x 2 shift types) in Table 10, the best performance is almost always obtained with a GDUQ variant: (accuracy: 8/8) as well as best calibration (6/8) or second best (2/8).

			Shift: Concept				Shift: Covariate			
			Accuracy (↑)		ECE (↓)		Accuracy (↑)		ECE (↓)	
Dataset	Domain	Calibration	No G-Δ UQ	G-Δ UQ	No G-Δ UQ	G-Δ UQ	No G-Δ UQ	G-Δ UQ	No G-Δ UQ	G-Δ UQ
		×	0.253±0.003	**0.281**±0.009	0.67±0.061	0.593±0.025	0.122±0.029	0.115±0.041	0.599±0.091	0.525±0.033
		CAGCN	0.253±0.005	0.268±0.008	0.452±0.14	0.473±0.12	0.122±0.018	0.092±0.161	0.355±0.227	0.396±0.161
		Dirichlet	0.229±0.018	0.22±0.022	0.472±0.06	0.472±0.03	0.244±0.105	**0.295**±0.044	**0.299**±0.092	0.328±0.044
		ETS	0.253±0.005	0.273±0.012	0.64±0.06	0.575±0.019	0.121±0.021	0.084±0.027	0.539±0.112	0.499±0.027
WebKB	University	GATS	0.253±0.005	0.273±0.01	0.608±0.008	0.485±0.02	0.122±0.018	0.079±0.029	0.455±0.057	0.376±0.029
		IRM	0.251±0.005	0.266±0.011	**0.342**±0.017	0.349±0.006	0.097±0.04	0.046±0.013	0.352±0.037	0.422±0.013
		Orderinvariant	0.253±0.005	0.27±0.01	0.628±0.026	0.564±0.024	0.122±0.018	0.106±0.065	0.545±0.079	0.47±0.065
		Spline	0.237±0.012	0.257±0.023	0.436±0.029	0.386±0.034	0.122±0.013	0.171±0.056	0.472±0.031	0.39±0.056
		VS	0.253±0.005	0.275±0.011	0.67±0.009	0.588±0.011	0.122±0.018	0.095±0.014	0.602±0.044	0.507±0.014
		×	0.581±0.003	0.595±0.003	0.307±0.009	0.13±0.011	0.47±0.002	0.518±0.014	0.348±0.032	0.141±0.008
		CAGCN	0.581±0.003	**0.597**±0.002	0.135±0.009	0.128±0.025	0.47±0.002	0.522±0.025	0.256±0.08	0.231±0.025
		Dirichlet	0.534±0.007	0.551±0.004	0.12±0.004	0.196±0.003	0.414±0.007	0.449±0.01	0.163±0.002	0.356±0.01
		ETS	0.581±0.003	0.596±0.004	0.301±0.009	0.116±0.018	0.47±0.002	**0.523**±0.003	0.31±0.077	0.141±0.003
Cora	Degree	GATS	0.581±0.003	0.596±0.004	0.185±0.018	0.229±0.039	0.47±0.002	0.521±0.011	0.211±0.004	0.308±0.011
		IRM	0.582±0.002	0.597±0.002	0.125±0.001	0.102±0.002	0.469±0.001	0.522±0.004	0.194±0.005	0.13±0.004
		Orderinvariant	0.581±0.003	0.592±0.002	0.226±0.024	0.213±0.049	0.47±0.002	0.498±0.027	0.318±0.042	0.196±0.027
		Spline	0.571±0.003	0.595±0.003	0.080±0.004	**0.068**±0.004	0.459±0.003	0.52±0.004	0.158±0.01	**0.098**±0.004
		VS	0.581±0.003	0.596±0.004	0.306±0.004	0.127±0.002	0.47±0.001	0.522±0.005	0.345±0.005	0.146±0.005
		×	0.607±0.003	0.628±0.001	0.284±0.009	0.111±0.013	0.603±0.004	0.633±0.031	0.263±0.004	0.118±0.019
		CAGCN	0.607±0.002	0.628±0.002	0.138±0.011	0.236±0.019	0.603±0.004	0.634±0.035	0.129±0.009	0.253±0.035
		Dirichlet	0.579±0.007	0.588±0.006	0.105±0.011	0.168±0.005	0.562±0.007	0.578±0.007	0.095±0.006	0.269±0.007
		ETS	0.607±0.002	0.628±0.002	0.282±0.002	0.11±0.003	0.603±0.004	0.634±0.013	0.243±0.023	0.106±0.013
Cora	Word	GATS	0.607±0.002	0.628±0.002	0.166±0.009	0.261±0.028	0.603±0.004	0.635±0.037	0.16±0.015	0.293±0.037
		IRM	0.608±0.001	0.63±0.002	0.115±0.002	0.088±0.003	0.602±0.003	0.635±0.004	0.106±0.002	0.098±0.004
		Orderinvariant	0.607±0.002	0.624±0.002	0.174±0.024	0.201±0.061	0.603±0.004	0.621±0.076	0.154±0.022	0.202±0.076
		Spline	0.598±0.005	0.629±0.002	0.073±0.002	**0.062**±0.005	0.591±0.002	0.635±0.004	0.063±0.006	**0.053**±0.004
		VS	0.607±0.001	**0.63**±0.002	0.283±0.003	0.111±0.003	0.603±0.004	**0.636**±0.003	0.261±0.005	0.119±0.003
		×	0.83±0.014	0.829±0.011	0.169±0.013	0.151±0.014	0.703±0.015	0.746±0.027	0.266±0.02	0.169±0.018
		CAGCN	0.83±0.013	0.83±0.013	0.137±0.011	0.143±0.022	0.703±0.019	0.749±0.033	0.25±0.021	0.186±0.017
		Dirichlet	0.801±0.02	0.806±0.008	0.161±0.012	0.17±0.01	0.671±0.018	0.771±0.03	0.241±0.029	0.217±0.017
		ETS	0.83±0.013	0.827±0.014	0.146±0.013	0.164±0.007	0.703±0.019	0.76±0.037	0.28±0.023	0.176±0.019
CBAS	Color	GATS	0.83±0.013	0.83±0.021	0.16±0.009	0.173±0.021	0.703±0.019	0.751±0.016	0.236±0.039	0.16±0.015
		IRM	0.829±0.013	0.839±0.015	0.142±0.009	**0.133**±0.006	0.72±0.019	0.803±0.04	0.207±0.035	**0.158**±0.017
		Orderinvariant	0.83±0.013	0.803±0.008	0.174±0.006	0.173±0.009	0.703±0.019	0.766±0.045	0.261±0.017	0.194±0.031
		Spline	0.82±0.016	0.824±0.011	0.159±0.009	0.16±0.014	0.683±0.019	0.786±0.038	0.225±0.034	0.179±0.035
		VS	0.829±0.012	**0.840**±0.011	0.166±0.011	0.146±0.012	0.717±0.019	**0.809**±0.008	0.242±0.019	0.182±0.014

Table C.10: **Layerwise Anchoring for Node Classification Datasets with Intermediate Representation Distributions.** Here, we provide preliminary results for performing layerwise anchoring when performing node classification. We fit a gaussian distribution over the representations (similar to node feature anchoring) and then sample anchors from this distribution. We fit a gaussian distribution over the representations (similar to node feature anchoring) and then sample anchors from this distribution. We see that these alternative strategies do provide benefits in some cases, but overall, our original input node feature anchoring strategy is more performant.

			Shift: Concept						Shift: Covariate					
				Accuracy (↑)			ECE (↓)			Accuracy (↑)			ECE (↓)	
Dataset	Domain	Calibration	No G-Δ UQ	Random 1	Random 2	No G-Δ UQ	Random 1	Random 2	No G-Δ UQ	Random 1	Random 2	No G-Δ UQ	Random 1	Random 2
CBAS	Color	Dirichlet	0.801±0.02	0.765±0.012	0.839±0.023	0.161±0.012	0.301±0.018	0.234±0.027	0.671±0.018	0.74±0.023	0.689±0.032	0.241±0.029	0.349±0.04	0.381±0.029
		ETS	0.83±0.013	0.819±0.012	0.82±0.088	0.146±0.013	0.23±0.017	0.257±0.021	0.703±0.019	0.638±0.051	0.686±0.026	0.28±0.023	0.347±0.037	0.334±0.028
		IRM	0.829±0.013	0.821±0.019	0.885±0.026	**0.142**±0.009	0.219±0.012	0.206±0.066	0.72±0.019	0.617±0.084	0.693±0.026	**0.207**±0.035	0.363±0.03	0.299±0.036
		Orderinvariant	0.83±0.013	0.813±0.015	0.819±0.028	0.174±0.006	0.255±0.015	0.236±0.006	0.703±0.019	0.831±0.008	0.636±0.026	0.261±0.017	0.286±0.039	0.303±0.062
		Spline	0.82±0.016	0.814±0.022	0.839±0.035	0.159±0.009	0.235±0.017	0.196±0.036	0.683±0.019	0.621±0.052	**0.757**±0.026	0.225±0.034	0.312±0.026	0.331±0.024
		VS	0.829±0.012	0.817±0.017	**0.91**±0.006	0.166±0.011	0.251±0.012	0.259±0.021	0.717±0.019	0.593±0.038	0.695±0.051	0.242±0.019	0.38±0.037	0.359±0.02
Cora	Degree	Dirichlet	0.534±0.007	0.483±0.014	0.423±0.007	0.12±0.004	0.355±0.004	0.347±0.004	0.414±0.007	0.466±0.073	0.425±0.005	0.163±0.002	0.315±0.042	0.345±0.007
		ETS	0.581±0.003	0.562±0.01	0.496±0.002	0.301±0.009	0.297±0.009	0.289±0.006	0.47±0.002	0.498±0.119	0.34±0.076	0.31±0.077	0.511±0.005	0.329±0.008
		IRM	**0.582**±0.002	0.567±0.011	0.492±0.003	0.125±0.001	**0.072**±0.003	0.116±0.006	0.469±0.001	0.499±0.117	0.508±0.005	0.194±0.005	**0.094**±0.009	0.105±0.006
		Orderinvariant	0.581±0.003	0.566±0.004	0.495±0.002	0.226±0.024	0.151±0.015	0.14±0.008	0.47±0.002	0.499±0.107	0.108±0.034	0.318±0.042	0.506±0.005	0.093±0.009
		Spline	0.571±0.003	0.561±0.011	0.493±0.005	0.080±0.004	0.11±0.01	0.119±0.005	0.459±0.003	0.499±0.12	0.508±0.006	0.158±0.01	0.105±0.03	0.127±0.012
		VS	0.581±0.003	0.571±0.002	0.279±0.009	0.306±0.004	0.493±0.008	0.272±0.009	0.47±0.001	**0.511**±0.091	0.51±0.002	0.345±0.005	0.347±0.051	0.323±0.007
Cora	Word	Dirichlet	0.579±0.007	0.581±0.004	0.504±0.004	0.105±0.011	0.271±0.011	0.285±0.002	0.562±0.007	0.586±0.009	0.497±0.01	0.095±0.006	0.264±0.022	0.275±0.007
		ETS	0.607±0.002	0.641±0.003	0.575±0.003	0.282±0.002	0.352±0.012	0.328±0.007	0.603±0.004	0.633±0.003	0.567±0.004	0.243±0.023	0.377±0.023	0.374±0.006
		IRM	0.608±0.001	**0.642**±0.002	0.574±0.003	0.115±0.002	0.106±0.004	0.154±0.005	0.602±0.003	0.635±0.004	0.569±0.003	0.106±0.002	0.136±0.012	0.173±0.007
		Orderinvariant	0.607±0.002	**0.642**±0.004	0.573±0.004	0.174±0.024	0.109±0.011	0.107±0.01	0.603±0.004	**0.638**±0.004	0.566±0.004	0.154±0.022	0.087±0.006	0.073±0.004
		Spline	0.598±0.005	0.641±0.002	0.576±0.004	0.073±0.002	0.076±0.004	**0.068**±0.007	0.591±0.002	0.632±0.002	0.568±0.003	**0.063**±0.006	0.066±0.005	0.077±0.004
		VS	0.607±0.001	0.639±0.003	0.583±0.005	0.283±0.003	0.345±0.007	0.335±0.012	0.603±0.004	0.637±0.004	0.579±0.004	0.261±0.005	0.396±0.028	0.384±0.005
WebKB	University	Dirichlet	0.229±0.018	0.214±0.000	0.228±0.012	0.472±0.06	0.56±0.000	0.552±0.041	0.244±0.105		0.347±0.012	0.299±0.092		0.429±0.05
		ETS	0.253±0.005	**0.279**±0.000	0.234±0.01	0.64±0.06	0.437±0.000	**0.33**±0.022	0.121±0.021		0.225±0.013	0.539±0.112		0.258±0.028
		IRM	0.251±0.005	0.251±0.000	0.232±0.009	0.342±0.017	0.379±0.000	0.459±0.01	0.097±0.04		0.187±0.021	0.352±0.037		0.294±0.018
		Orderinvariant	0.253±0.005	**0.279**±0.000	0.237±0.01	0.628±0.026	0.568±0.000	0.53±0.049	0.122±0.018		0.221±0.026	0.545±0.079		0.321±0.061
		Spline	0.237±0.012	0.237±0.000	0.233±0.008	0.436±0.029	0.467±0.000	0.483±0.041	0.122±0.013		0.205±0.01	0.472±0.031		0.329±0.035
		VS	0.253±0.005	**0.279**±0.000	0.234±0.01	0.67±0.009	0.49±0.000	0.344±0.02	0.122±0.018		0.201±0.011	0.602±0.044		0.256±0.014

Table C.11: **Layerwise Anchoring for Node Classification Datasets with Random Shuffling.** Here, we provide preliminary results for performing layerwise anchoring when performing node classification. We use random shuffling (similar to the proposed hidden layer strategy) to create the interemediate representations. We see that these alternative strategies do provide benefits.

			Shift: Concept						Shift: Covariate					
				Accuracy (↑)			ECE (↓)			Accuracy (↑)			ECE (↓)	
Dataset	Domain	Calibration	No G-Δ UQ	Batch 1	Batch 2	No G-Δ UQ	Batch 1	Batch 2	No G-Δ UQ	Batch 1	Batch 2	No G-Δ UQ	Batch 1	Batch 2
		Dirichlet	0.801±0.02	0.757±0.045	0.58±0.046	0.161±0.012	0.309±0.059	0.431±0.033	0.671±0.018	0.548±0.035	0.629±0.019	0.241±0.029	0.48±0.03	0.407±0.01
CBAS	Color	ETS	**0.83**±0.013	0.699±0.036	0.637±0.014	0.146±0.013	0.265±0.013	0.258±0.015	0.703±0.019	0.562±0.087	0.507±0	0.28±0.023	0.37±0.021	0.333±0.02
		IRM	0.829±0.013	0.711±0.031	0.724±0.029	**0.142**±0.009	0.284±0.032	0.291±0.02	**0.72**±0.019	0.59±0.079	0.657±0.037	**0.207**±0.035	0.336±0.032	0.268±0.037
		Orderinvariant	**0.83**±0.013	0.788±0.007	0.574±0.051	0.174±0.006	0.268±0.023	0.208±0.055	0.703±0.019	0.61±0.011	0.5±0.019	0.261±0.017	0.334±0.035	0.249±0.037
		Spline	0.82±0.016	0.695±0.039	0.652±0.022	0.159±0.009	0.279±0.018	0.236±0.013	0.683±0.019	0.49±0.124	0.6±0.032	0.225±0.034	0.364±0.034	0.308±0.054
		VS	0.829±0.012	0.73±0.043	0.693±0.051	0.166±0.011	0.264±0.009	0.197±0.033	0.717±0.019	0.429±0.083	0.607±0.042	0.242±0.019	0.478±0.042	0.312±0.014
		Dirichlet	0.534±0.007	0.515±0.003	0.442±0.012	0.12±0.004	0.304±0.01	0.315±0.004	0.414±0.007	0.507±0.004	0.419±0.006	0.163±0.002	0.28±0.006	0.338±0.004
Cora	Degree	ETS	0.581±0.003	0.576±0.011	0.516±0.013	0.301±0.009	0.317±0.018	0.285±0.007	0.47±0.002	0.563±0.003	0.496±0.005	0.31±0.077	0.373±0.009	0.311±0.006
		IRM	**0.582**±0.002	0.579±0.009	0.523±0.008	0.125±0.001	**0.076**±0.004	0.129±0.004	0.469±0.001	0.562±0.004	0.494±0.004	0.194±0.005	**0.088**±0.011	0.098±0.003
		Orderinvariant	0.581±0.003	**0.582**±0.003	0.518±0.005	0.226±0.024	0.134±0.023	0.126±0.012	0.47±0.002	0.561±0.004	0.496±0.004	0.318±0.042	0.091±0.014	0.096±0.007
		Spline	0.571±0.003	0.58±0.006	0.518±0.011	0.080±0.004	0.093±0.007	0.092±0.007	0.459±0.003	**0.565**±0.004	0.496±0.005	0.158±0.01	0.091±0.009	0.128±0.012
		VS	0.581±0.003	0.581±0.005	0.529±0.005	0.306±0.004	0.313±0.006	0.294±0.004	0.47±0.001	0.562±0.005	0.498±0.008	0.345±0.005	0.368±0.016	0.308±0.003
		Dirichlet	0.579±0.007	0.575±0.004	0.491±0.013	0.105±0.011	0.28±0.007	0.282±0.012	0.562±0.007	0.586±0.009	0.507±0.006	0.095±0.006	0.264±0.022	0.249±0.007
Cora	Word	ETS	0.607±0.002	0.636±0.003	0.562±0.006	0.282±0.002	0.359±0.02	0.311±0.006	0.603±0.004	0.633±0.003	0.561±0.005	0.243±0.023	0.377±0.023	0.365±0.005
		IRM	0.608±0.001	0.632±0.004	0.562±0.006	0.115±0.002	0.124±0.006	0.16±0.005	0.602±0.003	0.635±0.004	0.557±0.006	0.106±0.002	0.136±0.012	0.176±0.007
		Orderinvariant	0.607±0.002	**0.639**±0.003	0.561±0.006	0.174±0.024	0.111±0.008	0.095±0.006	0.603±0.004	**0.638**±0.004	0.56±0.004	0.154±0.022	0.087±0.006	0.076±0.006
		Spline	0.598±0.005	0.633±0.004	0.561±0.007	0.073±0.002	0.077±0.005	**0.069**±0.004	0.591±0.002	0.632±0.002	0.56±0.006	**0.063**±0.006	0.066±0.005	0.08±0.004
		VS	0.607±0.001	0.633±0.006	0.574±0.007	0.283±0.003	0.368±0.009	0.32±0.005	0.603±0.004	0.637±0.004	0.573±0.008	0.261±0.005	0.396±0.028	0.373±0.006
		Dirichlet	0.229±0.018	0.231±0.015	0.234±0.007	0.472±0.06	0.562±0.014	0.534±0.022	0.244±0.105	0.242±0.166	**0.298**±0.077	0.299±0.092	0.468±0.092	0.483±0.055
WebKB	University	ETS	0.253±0.005	0.277±0.007	0.234±0.003	0.64±0.06	0.421±0.017	0.327±0.015	0.121±0.021	0.128±0.017	0.101±0.033	0.539±0.112	0.437±0.032	**0.293**±0.01
		IRM	0.251±0.005	0.265±0.019	0.232±0.014	0.342±0.017	0.377±0.015	0.438±0.015	0.097±0.04	0.118±0.033	0.093±0.034	0.352±0.037	0.482±0.02	0.435±0.016
		Orderinvariant	0.253±0.005	0.268±0.01	0.231±0.01	0.628±0.026	0.513±0.071	0.431±0.025	0.122±0.018	0.122±0.018	0.1±0.029	0.545±0.079	0.475±0.049	0.38±0.069
		Spline	0.237±0.012	0.242±0.01	0.228±0.014	0.436±0.029	0.415±0.042	0.484±0.035	0.122±0.013	0.129±0.024	0.097±0.013	0.472±0.031	0.478±0.033	0.425±0.013
		VS	0.253±0.005	**0.279**±0.007	0.232±0.005	**0.67**±0.009	0.441±0.021	0.323±0.015	0.122±0.018	0.132±0.01	0.101±0.033	0.602±0.044	0.455±0.041	0.297±0.008

Table C.12: **Alternative Anchoring Strategies.** Here, we consider an alternative anchoring formulation for graph classification. Namely, instead of shuffling features across the batch (denoted Batch in the table), we perform READOUT anchoring by fitting a normal distribution over the hidden representations. We then randomly sample from this distribution to create anchors. Conceptually, this is similar to the node feature anchoring strategy. One potential direction of future work that is permitted by this formulation is to optimize the parameters of this distribution given a signal from an appropriate auxiliary task or loss. For example, we could perform an alternating optimization where the GNN is trained to minimize the loss, and the mean and variance of the anchoring distribution are optimized to minimize the expected calibration error on a separate calibration dataset. While a rigorous formulation is left to future work, we emphasize that the potential for improving the anchoring distribution, and thus controlling corresponding hypothesis diversity, is in fact a unique benefit of G-ΔUQ.

		Test Acc			*Test Cal*			*OOD Acc*			*OOD Cal*		
Shift Type	***Method***	***MPNN***	***Batch***	***Random***	***MPNN***	***Batch***	***Random***	***MPNN***	***Batch***	***Random***	***MPNN***	***Batch***	***Random***
GoodMotif, basis, concept	Dirichlet	0.995 ± 0.0007	0.994 ± 0.0002	0.996 ± 0.0009	0.040 ± 0.0037	0.036 ± 0.0016	0.035 ± 0.0058	0.924 ± 0.0069	0.923 ± 0.0117	0.942±0.0034	0.080 ± 0.0153	0.102 ± 0.0071	**0.062** ± 0.0086
	ETS	0.995 ± 0.0007	0.995 ± 0.0005	0.996 ± 0.0007	0.035 ± 0.0034	0.036 ± 0.0101	0.032 ± 0.0052	0.925 ± 0.0095	0.926 ± 0.009	0.935 ± 0.0068	0.095 ± 0.0098	0.096 ± 0.0128	0.087 ± 0.01451
	IRM	0.9954 ± 0.0007	0.9957 ± 0.0009	**0.9965** ± 0.0004	**0.0198** ± 0.0089	0.0229 ± 0.0105	0.0225±0.0038	0.9251 ± 0.0096	0.9301 ± 0.0123	**0.9462** ± 0.0024	0.0873 ± 0.0176	0.0966 ± 0.0103	0.0907 ± 0.0276
	OrderInvariant	0.995 ± 0.0007	0.995 ± 0.0005	0.995 ± 0.0005	0.033 ± 0.0094	0.028 ± 0.0047	0.032 ± 0.0009	0.925 ± 0.0095	0.928 ± 0.0104	0.935 ± 0.0027	0.090 ± 0.0092	0.093 ± 0.0070	0.0754±0.0029
	Spline	0.995 ± 0.0007	0.995 ± 0.0007	0.9962±0.0005	0.034 ± 0.0002	0.035 ± 0.0090	0.032 ± 0.0048	0.924 ± 0.0098	0.926 ± 0.0092	0.937 ± 0.0030	0.091 ± 0.0084	0.089 ± 0.0123	0.083 ± 0.0065
	VS	0.995 ± 0.0007	0.995 ± 0.0005	0.996 ± 0.000	0.035 ± 0.0034	0.036 ± 0.0087	0.033 ± 0.0098	0.925 ± 0.0094	0.926 ± 0.0095	0.936 ± 0.0053	0.094 ± 0.0096	0.095 ± 0.0133	0.082 ± 0.009
GoodMotif,basis, covariate	Dirichlet	0.999 ± 0.0003	0.999 ± 0.0004	0.999 ± 0.0002	0.017 ± 0.0054	0.017 ± 0.0019	0.014 ± 0.0004	0.685 ± 0.0504	0.650 ± 0.0450	**0.698** ± 0.0139	0.336 ± 0.0667	0.371 ± 0.0474	0.320 ± 0.0140
	ETS	0.9997±0.0004	0.999 ± 0.0005	0.999 ± 0.0002	0.0095±0.0091	0.017 ± 0.0064	0.017 ± 0.0056	0.690 ± 0.0434	0.649 ± 0.0476	0.686 ± 0.0226	0.313 ± 0.0413	**0.3739** ± 0.0485	0.334 ± 0.0167
	IRM	0.9997±0.0004	0.999 ± 0.0006	0.999 ± 0.0003	**0.0085** ± 0.0032	0.010 ± 0.0032	0.014 ± 0.0042	0.690 ± 0.0434	0.647 ± 0.0472	0.692 ± 0.0226	0.315 ± 0.0505	0.354 ± 0.0450	0.328 ± 0.0211
	OrderInvariant	0.9997±0.0004	0.999 ± 0.0005	0.999 ± 0.0003	0.014 ± 0.0028	0.020 ± 0.0090	0.013 ± 0.0081	0.690 ± 0.0434	0.649 ± 0.0450	0.689 ± 0.0170	0.320 ± 0.0501	0.358 ± 0.0410	0.328 ± 0.0218
	Spline	0.9997±0.0004	0.999 ± 0.0005	0.999 ± 0.0003	0.016 ± 0.0049	0.017 ± 0.0053	0.017 ± 0.0052	0.690 ± 0.0434	0.649 ± 0.0476	0.6923±0.0199	0.324 ± 0.0548	0.3733±0.0507	0.327 ± 0.0105
	VS	**0.9998** ± 0.0001	0.999 ± 0.0003	0.999 ± 0.0002	0.011 ± 0.0053	0.014 ± 0.0034	0.012 ± 0.0016	0.682 ± 0.0561	0.650 ± 0.0546	0.682 ± 0.0251	0.325 ± 0.0568	0.371 ± 0.0591	0.337 ± 0.0264
GOODSST2,length,concept	Dirichlet	0.938 ± 0.0019	0.939 ± 0.0056	**0.942** ± 0.00180	0.189 ± 0.01989	**0.165** ± 0.0179	0.187±0.0256	**0.694** ± 0.0193	0.693 ± 0.0020	0.687 ± 0.0027	0.146±0.0196	**0.133** ± 0.015	0.169 ± 0.0168
	ETS	0.938 ± 0.0020	0.939 ± 0.0060	0.941 ± 0.0017	0.389 ± 0.0018	0.390 ± 0.0022	0.393 ± 0.0007	0.6940±0.0193	0.692 ± 0.0019	0.687 ± 0.0034	0.214 ± 0.0098	0.216 ± 0.0033	0.220 ± 0.0057
	IRM	0.939 ± 0.0016	0.939 ± 0.0058	0.941 ± 0.0018	0.326 ± 0.0011	0.326 ± 0.0013	0.327 ± 0.0017	0.693 ± 0.0185	0.692 ± 0.0026	0.685 ± 0.0026	0.240 ± 0.0017	0.232 ± 0.0050	0.242 ± 0.0053
	OrderInvariant	0.938 ± 0.0020	0.939 ± 0.0060	0.941 ± 0.0022	0.314 ± 0.0014	0.315 ± 0.0029	0.315 ± 0.0012	0.6940±0.0193	0.692 ± 0.0019	0.687 ± 0.0033	0.224 ± 0.0010	0.222 ± 0.0030	0.223 ± 0.0054
	Spline	0.938 ± 0.0026	0.938 ± 0.0044	0.941 ± 0.0010	0.329 ± 0.0021	0.329 ± 0.0019	0.328 ± 0.0012	0.692 ± 0.0190	0.692 ± 0.0022	0.687 ± 0.0035	0.234 ± 0.0052	0.231 ± 0.0044	0.243 ± 0.0034
	VS	0.938 ± 0.0027	0.939 ± 0.0057	0.941±0.0018	0.290 ± 0.2099	0.484 ± 0.0008	0.487 ± 0.0007	0.693 ± 0.0184	0.693 ± 0.0018	0.687 ± 0.0031	0.331 ± 0.0484	0.375 ± 0.0022	0.382 ± 0.0048
GOODSST2,length,covariate	Dirichlet	**0.896** ± 0.0029	0.893 ± 0.0009	0.895 ± 0.00095	0.196 ± 0.0155	**0.172** ± 0.0091	0.1797±0.0109	0.825 ± 0.0037	0.827 ± 0.0066	0.805 ± 0.0150	0.163 ± 0.0198	**0.141** ± 0.0087	0.142±0.0122
	ETS	**0.8966** ± 0.0023	0.894 ± 0.0011	0.894 ± 0.0006	0.357 ± 0.0013	0.359 ± 0.0004	0.362 ± 0.0019	0.826 ± 0.0036	0.828±0.0065	0.806 ± 0.0117	0.309 ± 0.0050	0.314 ± 0.0076	0.300 ± 0.0070
	IRM	0.895 ± 0.0019	0.893 ± 0.0003	0.894 ± 0.0007	0.307 ± 0.0004	0.307 ± 0.0003	0.306 ± 0.0020	0.826 ± 0.0040	**0.828** ± 0.0065	0.809 ± 0.0152	0.276 ± 0.0046	0.277 ± 0.0061	0.265 ± 0.0078
	OrderInvariant	**0.896** ± 0.0023	0.894 ± 0.0011	0.894 ± 0.0008	0.288 ± 0.0008	0.285 ± 0.0008	0.284 ± 0.0013	0.826 ± 0.0036	0.828±0.0065	0.806 ± 0.0106	0.244 ± 0.0022	0.241 ± 0.0037	0.225 ± 0.0054
	Spline	0.894 ± 0.0016	0.890 ± 0.0009	0.892 ± 0.0040	0.309 ± 0.0024	0.307 ± 0.0009	0.307 ± 0.0022	0.822 ± 0.0026	0.822 ± 0.0092	0.801 ± 0.0110	0.275 ± 0.0043	0.276 ± 0.0063	0.264 ± 0.0063
	VS	0.8963±0.0028	0.893 ± 0.0008	0.894 ± 0.0007	0.291 ± 0.1833	0.460 ± 0.0011	0.465 ± 0.0010	0.821 ± 0.0053	0.827 ± 0.0071	0.806 ± 0.0119	0.299 ± 0.1395	0.431 ± 0.0061	0.429 ± 0.0054

C.9 Post-hoc Calibration Strategies

Several post hoc strategies have been developed for calibrating the predictions of a model. These have the advantage of flexibility, as they operate only on the outputs of a model and do not require that any changes be made to the model itself. Some methods include:

- **Temperature scaling (TS)** [34] simply scales the logits by a temperature parameter $T > 1$ to smooth the predictions. The scaling parameter T can be tuned on a validation set.
- **Ensemble temperature scaling (ETS)** [172] learns an ensemble of temperature-scaled predictions with uncalibrated predictions ($T = 1$) and uniform probabilistic outputs ($T = \infty$).
- **Vector scaling** (VS) [34] scales the entire output vector of class probabilities, rather than just the logits.
- **Multi-class isotonic regression (IRM)** [172] is a multiclass generalization of the famous isotonic regression method [278]): it ensembles predictions and labels, then learns a monotonically increasing function to map transformed predictions to labels.
- **Order-invariant calibration** [279] uses a neural network to learn an intra-order-preserving calibration function that can preserve a model's top-k predictions.
- **Spline** calibration instead uses splines to fit the calibration function [170].
- **Dirichlet calibration** [171] models the distribution of outputs using a Dirichlet distribution, using simple log-transformation of the uncalibrated probabilities which are then passed to a regularized fully connected neural network layer with softmax activation.

For node classification, some graph-specific post-hoc calibration methods have been proposed. **CaGCN** [41] uses the graph structure and an additional GCN to produce node-wise temperatures. GATS [42] extends this idea by using graph attention to model the influence of neighbors' temperatures when learning node-wise temperatures. We use the post hoc calibration baselines provided by **(author?)** in our experiments.

All of the above methods, and others, may be applied to the output of any model including one using G-ΔUQ. As we have shown, applying such post hoc methods to the outputs of the calibrated models may improve uncertainty estimates even more. Notably, calibrated models are expected to produce confidence estimates that match the true probabilities of the classes being predicted [280, 34, 169]. While poorly calibrated CIs are over/under confident in their predictions, calibrated CIs are more trustworthy and can also improve performance on other safety-critical tasks which implicitly require reliable prediction probabilities (see Sec. 5.5). We report the top-1 label expected calibration error (ECE) [281, 282]. Formally, let p_i be the top-1 probability, c_i be the predicted confidence, b_i a uniformly sized bin in $[0, 1]$. Then,

$$ECE := \sum_{i}^{N} b_i \|(p_i - c_i)\|$$

.

C.10 Details on Generalization Gap Prediction

Accurate estimation of the expected generalization error on unlabeled datasets allows models with unacceptable performance to be pulled from production. To this end, generalization error predictors (GEPs) [283, 284, 61, 285, 286] which assign sample-level scores, $S(x_i)$ which are then aggregated into dataset-level error estimates, have become popular. We use maximum softmax probability and a simple thresholding mechanism as the GEP (since we are interested in understanding the behavior of confidence indicators), and report the error between the predicted and true target dataset accuracy: $GEPError := ||\mathrm{Acc}_{target} - \frac{1}{|X|}\sum_i \mathbb{I}(\mathrm{S}(\bar{\mathrm{x}}_i; \mathrm{F}) > \tau)||$ where τ is tuned by minimizing GEP error on the validation dataset. We use the confidences obtained by the different baselines as sample-level scores, $\mathrm{S}(\mathrm{x}_i)$ corresponding to the model's expectation that a sample is correct. The MAE between the estimated error and true error is reported on both in- and out-of -distribution test splits provided by the GOOD benchmark.

C.11 Results on Generalization Error Prediction

GEP Experimental Setup. GEPs [283, 284, 61, 285, 286] aggregate sample-level scores capturing a model's uncertainty about the correctness of a prediction into dataset-level error estimates. Here, we use maximum softmax probability for scores and a thresholding mechanism as the GEP. (See Appendix C.10 for more details.) We consider `READOUT` anchoring with both pretrained and end-to-end training, and report the mean absolute error between the predicted and true target dataset accuracy on the OOD test split.

GEP Results. As shown in Table C.13, both pretrained and end-to-end G-ΔUQ outperform the vanilla model on 7/8 datasets. Notably, we see that pretrained G-ΔUQ is particularly effective as it obtains the best performance across 6/8 datasets. This not only highlights its utility as a flexible, light-weight strategy for improving uncertainty estimates without sacrificing accuracy, but also emphasizes that importance of structure, in lieu of full stochasticity, when estimating GNN uncertainties.

Table C.13: **GOOD-Datasets, Generalization Error Prediction Performance**. The MAE between the predicted and true test error on the OOD test split is reported. G-ΔUQ variants outperform vanilla models on 7/8 datasets (GOODMotif(Basis,Covariate) being the exception). Pretrained G-ΔUQ is particularly effective at this task as it achieves the best performance overall on 6/8 datasets. Promisingly, we see that regular G-ΔUQ improves performance over the vanilla model on 6/8 datasets (even if it is not the best overall). We further observe that performing generalization error prediction is more challenging under covariate shift than concept shift on the GOODCMNIST, GOODMotif(Basis) and GOODMotif(Size) datasets. On these datasets, the MAE is almost twice as large than their respective concept shift counterparts, across methods. GOODSST2 is the exception, where concept shift is in fact more challenging. To the best our knowledge, we are the first to investigate generalization error prediction on GNN-based tasks under distribution shift. Understanding this behavior further is an interesting direction of future work.

	CMNIST (Color)		**MotifLPE (Basis)**		**MotifLPE (Size)**		**SST2**	
Method	**Concept**(↓)	**Covariate** (↓)	**Concept**(↓)	**Covariate**(↓)	**Concept**(↓)	**Covariate**(↓)	**Concept**(↓)	**Covariate**(↓)
Vanilla	0.200 ± 0.009	0.510 ± 0.089	0.045 ± 0.003	$\mathbf{0.570 \pm 0.012}$	0.324 ± 0.018	0.537 ± 0.146	0.117 ± 0.006	0.056 ± 0.044
G-ΔUQ	$\mathbf{0.190 \pm 0.010}$	0.493 ± 0.072	0.023 ± 0.003	0.572 ± 0.019	0.317 ± 0.007	0.528 ± 0.189	0.124 ± 0.016	0.054 ± 0.043
Pretr. G-ΔUQ	0.192 ± 0.005	$\mathbf{0.387 \pm 0.048}$	$\mathbf{0.018 \pm 0.012}$	0.573 ± 0.004	$\mathbf{0.307 \pm 0.016}$	$\mathbf{0.356 \pm 0.143}$	$\mathbf{0.114 \pm 0.004}$	$\mathbf{0.030 \pm 0.026}$

C.12 Additional Study on Pretrained Variant

For the datasets and data shifts on which we reported out-of-distribution calibration error of pretrained vs. in-training G-ΔUQ earlier in Fig. 5.5, we now report additional results for in-distribution and out-of distribution accuracy as well as calibration error. We also include results for the additional GOODMotif-basis benchmark for completeness, noting that the methods provided by the original benchmark [35] generalized poorly to this split (which may be related to why G-ΔUQ methods offer little improvement over the vanilla model.) Fig. C.4 shows these extended results. By these additional metrics, we again see the competitiveness of applying G-ΔUQ to a pretrained model versus using it in end-to-end training.

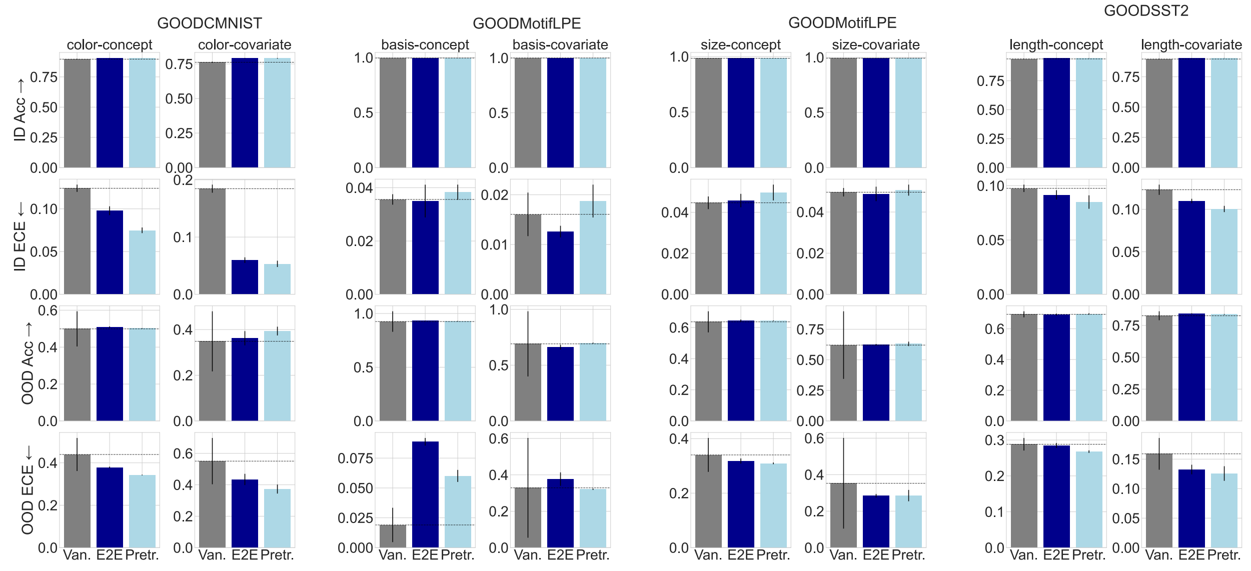


Figure C.4: **Evaluating Pretrained G-ΔUQ.** Here, we report the performance of pretrained G-ΔUQ models vs. end-to-end and vanilla models with respect to in-distribution and out-of-distribution accuracy as well as expected calibration error. With the exception of the GOODMotif (basis) dataset, pretrained G-ΔUQ improves the OOD ECE over both the vanilla model and end-to-end G-ΔUQ at comparable or improved OOD accuracy on 7/8 datasets. Furthermore, pretrained G-ΔUQ also improves the ID ECE on all but the GOODMotif (size) datasets (6/8), where it performs comparably to the vanilla model, and maintains the ID accuracy. (We note that all methods are comparably better calibrated on the GOODMotif ID data than GOODCMIST/GOODSST2 ID data; we suspect this is because there may exist simple shortcuts available in the GOODMotif dataset that can be used on the ID test set effectively.) Overall, these results clearly demonstrate that pretrained G-ΔUQ does offer some performance advantages over end-to-end G-ΔUQ and does so at reduced training times (see Table. C.13). For example, on GOODCMNIST (covariate shift), pretrained G-ΔUQ is not only 50% faster than end-to-end G-ΔUQ , it also improves OOD accuracy and OOD ECE over both the vanilla and end-to-end G-ΔUQmodels.

C.13 Runtime Table

Table C.14: **Runtimes.** We include the runtimes of both training per epoch (in seconds) and performing calibration. Reducing stochasticity can help reduce computation (L1 $\rightarrow$ L3). Cost can also be reduced by using a pretrained model.

	GOODCMNIST		GOODSST2		GOODMotifLPE	
Dataset	Training (S)	Inference (S)	Training (S)	Inference (S)	Training (S)	Inference (S)
Vanilla	18.5	25.8	10.8	18.5	3.8	4.5
Temp. Scaling	18.5	23.5	10.8	13.4	3.8	5.3
DEns (Ens Size=3)	18.456 x Ens Size	59.4	10.795 x Ens Size	29.0	3.8 x Ens Size	11.8
G-ΔUQ (L1, 10 anchors)	22.1	181.5	15.9	17.1	5.8	15.5
G-ΔUQ (L2, 10)	22.4	148.6	12.7	15.5	5.8	11.8
G-ΔUQ (HiddenRep, 10)	18.5	28.0	13.8	19.6	3.9	6.5
G-ΔUQ (Pretr. HiddenRep, 10)	8.6	27.8	6.8	16.0	2.5	6.4

C.14 Mean and Variance of Node Feature Gaussians

Table C.15: **Mean and Variance of Node Feature Anchoring Gaussians.** We report the mean and variance of the Gaussian distributions fitted to the input node features. Because the input node features vary in size, we report aggregate statistics over the mean and variance corresponding to each dimension. For example, Min(Mu) indicates that we are reports the minimum mean over the *d*-dim set of means.

Dataset	Domain	Shift	Min (Mu)	Max (Mu)	Mean (Mu)	Std (Mu)	Min (Std)	Max (Std)	Mean (Std)	Std (Std)
GOODSST2	length	concept	-4.563	0.69	-0.011	0.278	0.163	0.803	0.242	0.049
GOODSST2	length	covariate	-4.902	0.684	-0.01	0.3	0.175	0.838	0.255	0.05
GOODCMNIST	color	concept	0.117	0.133	0.127	0.008	0.092	0.097	0.095	0.003
GOODCMNIST	color	covariate	0.087	0.131	0.102	0.025	0.108	0.109	0.108	0
GOODMotifLPE	size	covariate	0.003	0.021	0.011	0.008	0.835	1.728	1.248	0.377
GOODMotifLPE	size	concept	-0.006	0	-0.002	0.003	0.542	1.114	0.783	0.242
GOODMotifLPE	basis	concept	-0.011	0.015	0.001	0.011	0.721	1.464	1.09	0.304
GOODMotifLPE	basis	covariate	-0.007	-0.002	-0.004	0.002	0.808	1.913	1.251	0.469
GOODWebKB	university	concept	0	0.95	0.049	0.099	0.001	0.5	0.168	0.095
GOODWebKB	university	covariate	0	0.934	0.05	0.104	0.001	0.5	0.164	0.098
GOODCora	degree	concept	0	0.507	0.007	0.017	0.001	0.5	0.061	0.051
GOODCora	degree	covariate	0	0.518	0.007	0.017	0.001	0.5	0.061	0.052
GOODCBAS	color	covariate	0.394	0.591	0.471	0.093	0.142	0.492	0.403	0.174
GOODCBAS	color	concept	0.23	0.569	0.4	0.144	0.168	0.495	0.39	0.152

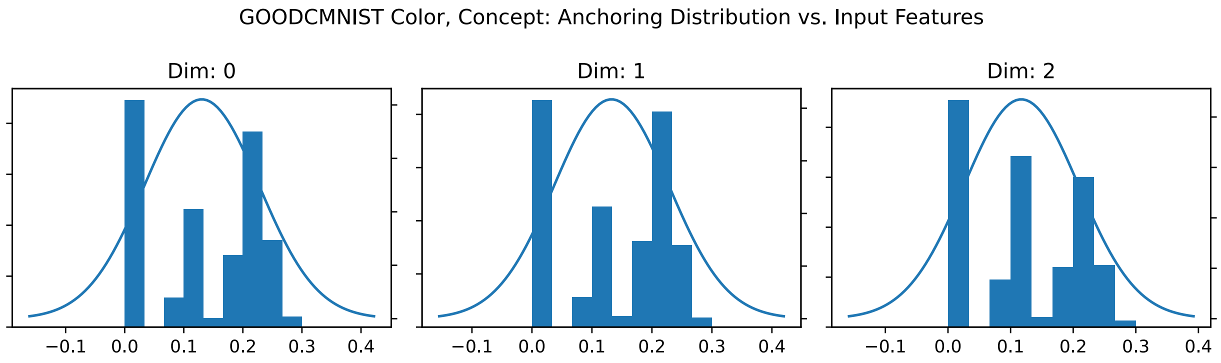


Figure C.5: **GOODCMNIST, Concept, Anchoring Distribution.** We plot the mean and variance of the fitted anchoring distribution vs. the true feature distribution for each input dimension. We observe there is a mismatch between the empircal distribution and the fitted Gaussian. However, we did not find this mismatch to harm the effectiveness of G-ΔUQ.

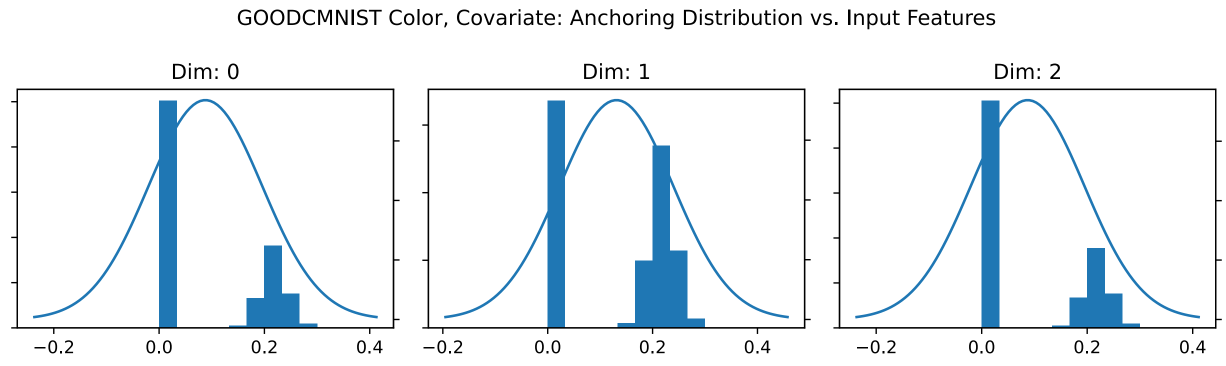


Figure C.6: **GOODCMNIST, Covariate, Anchoring Distribution.** We plot the mean and variance of the fitted anchoring distribution vs. the true feature distribution for each input dimension. We observe there is a mismatch between the empircal distribution and the fitted Gaussian. However, we did not find this mismatch to harm the effectiveness of G-ΔUQ.

Table C.16: **Number of Parmeters per Model.**We provide the number of parameters in the vanilla and modified parameter as follows. Note, that the change in parameters is architecture and input dimension dependent. For example, GOODCMNIST, and GOODSST2 use GIN MPNN layers. Therefore, when changing the layer dimension, we are changing the dimension of its internal MLP. It is not an error that intermediate layer G-ΔUQhave the same number of parameters, this is due to the architecture: these layers are the same size in the vanilla model. Likewise, GOODCora's input features have dimension is 8701, so doubling the input layer's dimension appears to add a signficant number of parameters. We do not believe this

Dataset	GOODCMNIST	GOODMotif	GOODSST2	GOODCORA	GOODWebKB	GOODCBAS
Baseline	2001310	911403	1732201	2816770	695105	185104
G-ΔUQ(NFA)	2003110	913803	2193001	5429770	1206005	186304
G-ΔUQ(L1)	2360110	1633203	2091001			
G-ΔUQ(L2)	2360110	1633203	2091001			
G-ΔUQ(L3)	2360110					
G-ΔUQ(L4)	2360110					
G-ΔUQ(Readout)	2004310	912303	1732501			

C.15 Expanded Discussion on Anchoring Design Choices

Below, we expand upon some of the design choices for the proposed anchoring strategies.

When performing node featuring anchoring, how does fitting a Gaussian distribution to the input node features help manage the combinatorial stochasticity induced by message passing?

Without loss of generality, consider a node classification setting, where every sample is assigned a unique anchor. Then, due to message passing, after l hops, a given node's representation will have aggregated information from its l hop neighborhood. However, since each node in this neighborhood has a unique anchor, we see that any given node's representation is not only stochastic due to its own anchor but also that of its neighbors. For example, if any of its neighbors are assigned a different anchor, then the given node's representation will change, even if its own anchor did not. Since this behavior holds true for all nodes and each of their respective neighborhoods, we loosely refer to this phenomenon having combinatorial complexity, as effectively marginalizing out the anchoring distribution would require handling any and all changes to all l-hop neighbors. In contrast, when performing anchored image classification, the representation of a sample is only dependent on its unique, corresponding anchor, and is not influenced by the anchors of other samples. To this end, using the fitted Gaussian distribution helps manage this complexity, since changes to the anchors of a node's l-hop neighborhood are simpler to model as they require only learning to marginalize out a Gaussian distribution (instead of the training distribution). Indeed, for example, if we were to assume simplified model where message passing only summed node neighbors, the anchoring distribution would remain Gaussian after l rounds of message passing since the sum of Gaussian is still Gaussian (the exact parameters of the distribution would depend on the normalization used however).

APPENDIX D

Graph Clustering with LLM Guidance

In this section, we briefly introduce deep attributed graph clustering and relevant works for combining LLMs and GNNs when working with TAGs. Please see [30] and [213], respectively, for comprehensive surveys.

Deep Attributed Graph Clustering. While unattributed graph clustering has a rich history in network analysis through modularity maximization, spectral clustering, and cuts-based approaches, the success of GNNs in graph representation learning has lead to growing interest in deep clustering methods that efficiently leverage both node-level attributes and topology. Broadly, such methods either (i) learn node representations using a self-supervised or unsupervised objective, and then perform clustering given these representations or (ii) learn both the embeddings and clustering assignments end-to-end through specialized clustering-based losses. While reconstructive [214, 215] and adversarial frameworks [216] were initially popular, in this work, we focus on contrastive [77, 217, 52, 78] and pooling-based methods [209, 207, 208]. Such methods, which, respectively, use contrastive losses to learn discriminative node representations or propose novel pooling layers that optimize for clustering-based losses (e.g., spectral relaxations of modularity or mincut), are more performative, efficient, and scalable than adversarial or reconstructive approaches. Moreover, as we will discuss in Sec. 7.4.2, these methods are more amenable to fine-tuning. Indeed, fine-tuning contrastively pre-trained representations is well-known to induce state-of-the-art performance on a variety of supervised tasks in both vision and graph representation learning.

LLMs + Graphs. Recent approaches that seek to combine graphs/GNNs and natural-language/LLMs can be categorized as being "predictors" (the LLM provides predictions), "encoders" (sentence transformers or other LLMs are used to provide input node features), or "aligners" (GNNs and LLMs jointly trained to perform the task) [213]. Various mechanisms, including prompting [218], fine-tuning [219], variational expectation maximization [67], joint optimization [9], and distillation [220], have been proposed to fulfill these roles, typically on supervised tasks. Instead, GCLR uses the LLM as a refiner and enhancer, as the LLM is only prompted to provide feedback for updating the underlying GNN-based graph clustering solution and sentence transformers are used to provide input node embeddings. This allows us to avoid the expensive fine-tuning of either LLMs or pre-trained language models, as well as exploit the scalability of graph clustering algorithms.

D.1 Additional Results

Table D.1: **Query Function Ablation.**We report performance on the following query strategies: random sampling \ entropy sampling \ least confidence \ margin sampling. We observe that while there is a slight decrease in performance when using random sampling as the query function, overall margin sampling perform similarly to entropy sampling. Least confidence sampling, in fact, improves performance on a few cases.

Dataset	**Method**	**Acc.**	**NMI**	**F1**	**ARI**	**COND**	**MOD**
citeseer	diffpool	49.45\59.56\60.19\59.38	26.47\27.75\28.38\23.21	8.47\13.16\12.88\8.74	33.87\31.93\31.16\29.23	0.16\0.11\0.09\0.1	0.39\0.34\0.33\0.29
	dinknet	46.14\56.99\58.16\57.9	6.62\35.41\35.8\36.2	2.81\22.96\22.28\22.96	10.65\37.79\37.39\40.21	0.02\0.22\0.22\0.21	0.07\0.43\0.43\0.44
	dmon	37.94\51.74\52.18\51.31	7.73\27.88\28.34\27.54	2.53\18.42\18.2\17.98	11.86\33.67\33.88\32.75	0.06\0.15\0.15\0.15	0.16\0.51\0.5\0.5
	mincut	64.08\63.4\63.56\61.16	34.45\34.74\35.16\39.89	30.4\31.31\31.42\30.13	55.48\54.43\55.27\49.44	0.24\0.23\0.23\0.31	0.56\0.58\0.58\0.52
cora	diffpool	68.7\66.53\66.55\66.52	43.99\45.88\45.32\45.52	36.35\42.32\42.02\41.83	55.24\54.14\53.08\53.96	0.32\0.26\0.26\0.26	0.5\0.53\0.53\0.53
	dinknet	35.33\42.7\42.4\42.67	14.46\26.75\26.51\26.92	10.65\19.01\18.61\19	11.3\30.6\30.01\30.12	0.08\0.39\0.38\0.37	0.13\0.29\0.29\0.29
	dmon	46.14\56.99\58.16\57.9	6.62\35.41\35.8\36.2	2.81\22.96\22.28\22.96	10.65\37.79\37.39\40.21	0.02\0.22\0.22\0.21	0.07\0.43\0.43\0.44
	mincut	61.16\61.52\60.5\60.61	39.89\40.94\39.78\40.05	30.13\29.34\29\30.1	49.44\50.6\50.34\50.5	0.31\0.31\0.33\0.32	0.52\0.51\0.5\0.5
wikics	diffpool	37.94\51.74\52.18\51.31	7.73\27.88\28.34\27.54	2.53\18.42\18.2\17.98	11.86\33.67\33.88\32.75	0.06\0.15\0.15\0.15	0.16\0.51\0.5\0.5
	dinknet	64.08\63.4\63.56\61.78	34.45\34.74\35.16\33.38	30.4\31.31\31.42\29	55.48\54.43\55.27\54.02	0.24\0.23\0.23\0.25	0.56\0.58\0.58\0.57
	dmon	35.33\42.7\42.4\42.67	14.46\26.75\26.51\26.92	10.65\19.01\18.61\19	11.3\30.6\30.01\30.12	0.08\0.39\0.38\0.37	0.13\0.29\0.29\0.29
	mincut	46.4\46.92\44.46\45.76	22.06\18.61\20.11\18.19	11.57\5.6\6.79\9.08	28.09\22.16\22.67\22.37	0.27\0.24\0.28\0.21	0.22\0.16\0.2\0.16

Table D.2: **Ablation on the Labeling Budget.** We report performance when the LLM labeling budget is 20% \ 40% \ 60% \ 80% \ 100%. We find that increasing the budget does not substantially increase performance, unlike traditional active learning. We hypothesize this is partially due to regularizing training using GNN pseudo-labels and the imperfect LLM oracle.

Dataset	**Method**	**Acc.**	**NMI**	**F1**	**ARI**	**COND**	**MOD**
citeseer	diffpool	54.43\53.82\52.77\53.05\54.66	23.52\22.7\22.59\22.76\22.21	15.33\15.3\15.2\15.54\15.23	35.44\36.46\36.72\36.88\36.52	0.3\0.3\0.31\0.32\0.32	0.45\0.45\0.45\0.44\0.44
	dinknet	69.81\69.84\69.81\69.81\69.81	34.15\36.98\36.61\36.38\36.19	26.36\29.83\29.23\28.97\28.82	45.51\48.06\47.35\47.13\46.93	0.07\0.07\0.07\0.07\0.07	0.6\0.62\0.61\0.61\0.61
	dmon	51.81\51.63\51.62\51.3\51.25	28.17\29.15\28.99\29.19\29.1	18\18.82\18.37\18.65\18.56	31.84\32.72\32.66\32.53\32.45	0.13\0.14\0.15\0.15\0.15	0.5\0.5\0.5\0.5\0.5
	mincut	63.57\61.68\63.12\63.98\64.14	34.44\32.88\33.4\32.94\32.7	29.95\27.77\28.04\27.41\27.23	55.75\54.3\54.29\53.44\52.27	0.26\0.31\0.3\0.31\0.3	0.55\0.51\0.51\0.51\0.51
cora	diffpool	55.55\55.44\55.61\56.67\56.53	24.17\24.81\24.9\24.97\25.18	9.91\10.06\10.35\10.9\11.02	31.91\32.92\33.57\34.3\34.25	0.41\0.43\0.44\0.44\0.44	0.34\0.33\0.32\0.33\0.34
	dinknet	59.97\60.01\60.01\59.97\60.04	24.84\25.72\25.36\25.53\25.23	10.19\10.89\10.43\10.43\10	30.36\30.74\30.58\30.79\30.47	0.09\0.09\0.09\0.09\0.09	0.32\0.33\0.32\0.32\0.31
	dmon	58.14\58.89\59\58.91\58.91	35.71\36.31\36.95\37.39\37.5	22.31\21.97\21.82\21.85\22.25	38.78\37.71\37.93\39.24\39.44	0.21\0.22\0.21\0.2\0.19	0.43\0.43\0.44\0.44\0.45
	mincut	60.43\62.17\59.4\60.54\60.76	38.36\37.51\38.47\38.18\38.84	28.27\27.65\28.51\28.37\29.36	48.36\47.79\48.61\47.87\47.84	0.36\0.37\0.38\0.37\0.38	0.47\0.46\0.45\0.46\0.46
subtagwikics	diffpool	49.75\51.71\52.18\51.31\52.5	30.03\30.54\30.14\31.23\30.73	20.62\22.71\19.96\19.74\20.97	40.33\40.68\37.45\38.23\39.88	0.43\0.39\0.39\0.39\0.41	0.39\0.42\0.42\0.38\0.4
	dinknet	66.56\66.54\66.54\66.51\66.52	46.13\45.77\45.73\45.68\45.62	42.49\42.43\42.31\42.14\42.05	54.59\54.27\54.16\54.14\54.21	0.26\0.25\0.26\0.26\0.26	0.53\0.53\0.53\0.53\0.53
	dmon	42.99\42.83\42.9\43.05\43.13	27.31\27.7\27.83\28.09\28.32	19.24\19.35\19.57\19.73\19.97	30.42\30.68\30.76\31.06\31.12	0.37\0.37\0.37\0.36\0.36	0.29\0.29\0.3\0.3\0.3
	mincut	44.91\44.01\44.53\43.05\44.98	18.36\17.12\20.1\19.83\18.77	5.11\5.97\9.44\9.48\6.2	21.5\18.35\25.49\18.43\19.59	0.28\0.22\0.3\0.3\0.26	0.16\0.14\0.15\0.17\0.19

D.2 Prompt Examples

D.3 Metrics

We consider the following extrinsic and graph topology-based metrics in our evaluation. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{X}, [\mathcal{Y}])$ represent a graph with its respective node-set, edge-set, raw node based text information, embedded node attribute information (e.g., some embedding of a node's text), and optional ground-truth cluster assignment. Further, let N be the number of the nodes, M be the number of edges, C be the desired (or ground-truth) number of clusters, d the dimension of the hidden representation, $\mathbf{A} \in \mathbb{R}^{n \times n}$ be the corresponding adjacency matrix, $\mathbf{X} \in \mathbb{R}^{N \times d}$ be a matrix representation of $\mathcal{X}$, $\mathbf{Y} \in [0,1]^C$, $\mathbf{d}_v$ be the degree vector of a particular node v, and c_v be the *predicted* cluster of a given node v.

- **Modularity [202].** Modularity measures the deviation with respect to nodes belonging to the same cluster against the expectation of the nodes being connected given a null model where nodes are connected randomly. Graphs with high modularity will have clusters where the majority of the edges are contained with some cluster and few edges that cross the clusters. Modularity falls within $[-\frac{1}{2}, 1]$, where a positive score indicates that the clustering structure that is above random, and is defined as follows:

$$Q = \frac{1}{2m} \sum_{ij} \left[\mathbf{A}_{[ij]} - \frac{d_i d_j}{2m} \right] \mathbb{1}[c_i = c_j].$$

- **Conductance [204, 287].** Also known as the Cheeger coefficient, this metric measures how quickly a random walk on a graph will reach its stationary distribution. Given a particular cluster, $\hat{c}$, the number of edges belonging to that cluster (intra-cluster edges) can be computed as $r_{\hat{c}} = \sum_{u,v \in \mathbf{A}} \mathbb{1}[c_u = \hat{c}, c_v = \hat{c}]$, and the number of edges are not fully contained in $\hat{c}$ (inter-cluster edges) can be computed as $s_{\hat{c}} = \sum_{u,v \in \mathbf{A}} \mathbb{1}[c_u = \hat{c}, c_v \neq \hat{c}]$. Then, conductance is defined as the average ratio of intra- and inter- cluster edges, where tight clusters are expected to have relatively fewer inter cluster edges.

$$\phi = \frac{1}{C} \sum_{\hat{c}}^{C} \frac{s_{\hat{c}}}{r_{\hat{c}} + s_{\hat{c}}}$$

- **Accuracy.**

$$ACC = \sum_{i=1}^{n} \frac{\phi(y_i, map(\hat{y}_i))}{n} \tag{D.1}$$

$\hat{y}_i$ represents the predicted cluster ID, while y_i indicates the ground truth cluster ID

Table D.3: **Prompt Example: Concepts, CORA**

CONCEPTS GENERATION PROMPT: Task: I'm clustering papers in a citation network according to research area and need help coming up with cluster names. The following `num-exemplars` papers that have been clustered together and I'm going to give you their abstract/titles. Can you propose a < 7 word research topic and 2-3 sentence description for this cluster? Try not to make it too specific or too broad, and explain your reasoning. Return your answer in a JSON format: {topic: [your topic], description: [your description], reasoning: [your reasoning]}.

SAMPLES FROM CLUSTER:
```
Sample 1
Sample 2
...
Sample Num-Exemplars
```

Answer:

CONCEPT PREDICTION PROMPT:
[Task]
I'm currently working on clustering papers within a citation network based on their abstracts/titles. I'm seeking assistance in determining the cluster association for a specific uncertain sample. You'll be provided with the abstract/title of this sample, along with the titles and short descriptions of `num-clusters` potential clusters. Your task involves carefully reading each cluster title and description, taking a thoughtful approach, and selecting the cluster that best aligns with the confusing sample. Please provide your answer in JSON format, including the predicted cluster number, title of the predicted cluster, and your detailed reasoning. Your response should look like this: {cluster: [your predicted cluster number], cluster title: [title of predicted cluster], reasoning: [your reasoning for choosing this cluster]}. Take your time and ensure clarity in your explanation.

[CLUSTER TITLES]
1. `<GENERATED TITLE>`
Description: `<GENERATED TITLE DESCRIPTION>`

...
NUM-CLUSTERS. `<GENERATED TITLE>`
Description: `<GENERATED TITLE DESCRIPTION>`

[UNCERTAIN SAMPLE]
`QUERY`

[ANSWER]

Table D.4: **Prompt Example: Incontext, CORA**

PROMPT:
[Example]
`<Sample>`
{Category: `<GNN's Predicted Cluster>`}

...
[Example]
`<Sample>`
{Category: `<GNN's Predicted Cluster>`}

[Task]
Given the above examples, please identify the correct category for the following query sample. Please explain your reasoning and return your answer in a JSON format: category: [your prediction], reasoning: [your reasoning]. If you're unsure of an answer, select category -1.

[QUERY]
`<QUERY>`

[ANSWER]

Table D.5: **Prompt Example: Triplets, CORA**

PROMPT: Task: I'm clustering papers in a citation network according to research area and need help determining where a particular query sample belongs given its abstract and title. I will give you the abstracts/titles of two samples belonging to nearby clusters and you should select the abstract/title that is more similar to the query in terms of research topic. Please explain your reasoning and return your answer in a JSON format: {selection: [1,2,-1(neither or unsure)], reasoning: [your reasoning]}.

[SAMPLE 1]
`<Sample from 1st (2nd) Closest Cluster>]`

[SAMPLE 2]
`<Sample from 2nd (1st) Closest Cluster>]`

[QUERY]
`<Sample of Query Sample>]`

[ANSWER]

label. $map(.)$ denotes the Kuhn-Munkres algorithm [288] which aligns the predicted cluster-ID with the class-ID, and indicator function $\phi(.)$ is formulated as:

$$\phi(y_i, map(\hat{y}_i)) = \begin{cases} 1 & \text{if } y_i = map(\hat{y}_i) \\ 0 & \text{else} \end{cases} \tag{D.2}$$

- **Normalized Mutural Information.**

$$NMI = -\frac{2\sum_{\hat{y}}\sum_{y} p(\hat{y}, y) \log \frac{p(\hat{y},y)}{p(\hat{y})p(y)}}{\sum_i p(\hat{y_i}) \log(p(\hat{y_i})) + \sum_j p(y_j) \log(p(y_j))} \tag{D.3}$$

where $p(y), p(\hat{y}),$ and $p(\hat{y}, y)$ represent the distribution of predicted results, distribution of the ground truth, and joint distribution of them, respectively.

- **Adjusted Random Index.**

$$ARI = \frac{RI - expectedRI}{max(RI) - expectedRI} \tag{D.4}$$

where RI and $expectedRI$ signifies the Rand Index and expected Rand Index [289], respectively. An ARI of 0 suggests disagreement between real and modeled clustering in pairing, whereas an ARI of 1 indicates concordance between real and modeled clustering, representing identical clusters.

- **F1-Score.**

$$F1 = \frac{2.Precision.Recall}{Precision + Recall} \tag{D.5}$$

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN} \tag{D.6}$$

where TP, FP, and FN indicate the number of true positive, false positive, and false negative samples, respectively.

D.4 Reproducibility Statement

All code will be released upon acceptance. We dropped the computation linguistic and web-technology categories from WikiCS to create a more even and separate labeling for evaluation. We use the mixtral-8x-7b model, and a G.5 (8 gpu) instance on AWS. We repeat results over 3 seeds.

Table D.6: **Dataset Statistics.**

Dataset	Num Nodes	Num Edges	Num Clusters
Cora [290]	2,708	5,429	7
Citeseer [291]	3,327	4,732	6
WikiCS* [292]	10,601	204120	8

D.5 Example of Generated Titles

Table D.7: **Generated Concepts.** Below, are examples of concepts generated by `chatgpt-3.5-turbo` on Cora with MinCut as the GNN clustering algorithm. While some concepts are imperfect, e.g., rule learning or theory, other topics are well captured. Applying self-refinement strategies could improve these generated concepts, at additional budget expenditure.

True	Generated
Reinforcement Learning	Reinforcement Learning and Dynamic Programming
Genetic Algorithms	Evolutionary Algorithms in Problem Solving
Rule Learning	Error Bounds in Learning Algorithms
Theory	Feature Selection in Machine Learning'
Probabilistic Methods	Bayesian Statistical Methods
Case Based	Improving Case-Based Reasoning Adaptation
Neural Networks	Neural Network Self-Organization

BIBLIOGRAPHY

[1] Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2020.

[2] Giannis Nikolentzos, Antoine J.-P. Tixier, and Michalis Vazirgiannis. Message passing attention networks for document understanding. In *AAAI*, 2020.

[3] Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. Every document owns its structure: Inductive text classification via graph neural networks. In *Proc.Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, July 2020.

[4] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2013.

[5] Jason W. Wei and Kai Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proc.Conf.on Empirical Methods in Natural Language Processing and Int. Joint Conf.on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[6] SueYeon Chung, Daniel D. Lee, and Haim Sompolinsky. Classification and geometry of general perceptual manifolds. *Physical Review X*, 8(3), 2018.

[7] Jeff Z. HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. In *Proc.Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[8] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2020.

[9] Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*, 2023.

[10] Govindan Subramanian, Bharath Ramsundar, Vijay Pande, and Rajiah Aldrin Denny Denny. Computational modeling of β-secretase 1 (BACE-1) inhibitors using ligand based approaches. *Journal of Chemical Information and Modeling*, 2016.

[11] Mengying Sun, Jing Xing, Huijun Wang, Bin Chen, and Jiayu Zhou. MoCL: Data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph. In *Proc.ACM Int.Conf.on Knowledge Discovery & Data Mining (SIGKDD)*, 2021.

[12] Mucong Ding, Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Micah Goldblum, David Wipf, Furong Huang, and Tom Goldstein. A closer look at distribution shifts and out-of-distribution generalization on graphs. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

[13] Raphael Gontijo Lopes, Sylvia J. Smullin, Ekin D. Cubuk, and Ethan Dyer. Tradeoffs in data augmentation: An empirical study. In *Int.Conf.on Learning Representations (ICLR)*, 2020.

[14] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2020.

[15] Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. Openattack: An open-source textual adversarial attack toolkit. In *Proc. Assoc. Comp. Linguistics and Int. Joint Conf. on Natural Language Processing: System Demonstrations*, 2021.

[16] Vikas Verma, Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc V. Le. Towards domain-agnostic contrastive learning. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2021.

[17] Davide Buffelli, Pietro Liò, and Fabio Vandin. SizeShiftReg: A regularization method for improving size-generalization in graph neural networks. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2022.

[18] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2021.

[19] Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. Size-invariant graph representations for graph classification extrapolations. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2021.

[20] Marinka Zitnik, Rok Sosič, and Jure Leskovec. Prioritizing network communities. *Nature Communications*, 2018.

[21] Andrew J Dudzik and Petar Veličković. Graph neural networks are dynamic programmers. In *Proc. on Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[22] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2018.

[23] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proc.ACM SIGKDD Int.Conf.on Knowledge Discovery & Data Mining (KDD)*, 2018.

[24] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2017.

[25] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2018.

[26] Gabriele Corso, Luca Cavalleri, Dominique Beaini, and Pietro Liò. Principal neighbourhood aggregation for graph nets. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2020.

[27] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2019.

[28] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2017.

[29] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2015.

[30] Yue Liu, Jun Xia, Sihang Zhou, Siwei Wang, Xifeng Guo, Xihong Yang, Ke Liang, Wenxuan Tu, Stan Z. Li, and Xinwang Liu. A survey of deep graph clustering: Taxonomy, challenge, and application. abs/2211.12875, 2022.

[31] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. REGAL: representation learning-based graph alignment. In *Proceedings of Int. Conf. on Information and Knowledge Management, CIKM*, 2018.

[32] Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. Unsolved problems in ML safety. *CoRR*, abs/2109.13916, 2021.

[33] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2015.

[34] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proc.of the Int.Conf.on Machine Learning, (ICML)*, 2017.

[35] Shurui Gui, Xiner Li, Limei Wang, and Shuiwang Ji. GOOD: A graph out-of-distribution benchmark. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS), Benchmark Track*, 2022.

[36] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proc. ACM Int. Conf. on Knowledge Discovery & Data Mining, KDD*, 2018.

[37] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2018.

[38] Sirui Yao and Bert Huang. Beyond parity: Fairness objectives for collaborative filtering. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[39] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Controlling popularity bias in learning-to-rank recommendation. In *Proc. of ACM Conference on Recommender Systems (RecSys)*, 2017.

[40] Yibo Li, Xiao Wang, Yujie Xing, Shaohua Fan, Ruijia Wang, Yaoqi Liu, and Chuan Shi. Graph fairness learning under distribution shifts. In *Proc. of the ACM on Web Conference (WWW)*, 2024.

[41] Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. Be confident! Towards trustworthy graph neural networks via confidence calibration. In *Proc.Adv.in Neural Information Processing Systems NeurIPS*, 2021.

[42] Hans Hao-Hsun Hsu, Yuesong Shen, Christian Tomani, and Daniel Cremers. What makes graph neural networks miscalibrated? In *Proc.Adv.in Neural Information Processing Systems NeurIPS*, 2022.

[43] Lukas Gosch, Simon Geisler, Daniel Sturm, Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Adversarial training for graph neural networks: Pitfalls, solutions, and new directions. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[44] Yongqiang Chen, Yonggang Zhang, Yatao Bian, Han Yang, Kaili Ma, Binghui Xie, Tongliang Liu, Bo Han, and James Cheng. Learning causally invariant representations for out-of-distribution generalization on graphs. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[45] Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *Proc. Int. Conf. on Machine Learning, ICML*, 2022.

[46] Ching-Yao Chuang and Stefanie Jegelka. Tree mover's distance: Bridging graph metrics and stability of graph neural networks. In *Proc.Adv.in Neural Information Processing Systems NeurIPS*, 2022.

[47] Tianle Cai, Shengjie Luo, Keyulu Xu, Di He, Tie-Yan Liu, and Liwei Wang. GraphNorm: A principled approach to accelerating graph neural network training. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2021.

[48] Keyulu Xu, Mozhi Zhang, Stefanie Jegelka, and Kenji Kawaguchi. Optimization of graph neural networks: Implicit acceleration by skip connections and more depth. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2021.

[49] Wei Ju, Siyu Yi, Yifan Wang, Qingqing Long, Junyu Luo, Zhiping Xiao, and Ming Zhang. A survey of data-efficient graph learning. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI 2024)*, 2024.

[50] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proc. ACM Int. Conf. on Web Search and Data Mining (WSDM)*, 2021.

[51] Kaveh Hassani and Amir Hosein Khas Ahmadi. Contrastive multi-view representation learning on graphs. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2020.

[52] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Medhi Azabou, Eva Dyer, Rémi Munos, Petar Velickovic, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2022.

[53] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z. Li. SimGRACE: A simple framework for graph contrastive learning without data augmentation. In *Proc.Acm Conf.on World Wide Web (WWW)*, 2022.

[54] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2020.

[55] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proc.Int.Conf.on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[56] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, and Piotr Bojanowski. Unsupervised learning of visual features by contrasting cluster assignments. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2020.

[57] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2019.

[58] Hong Liu, Jeff Z. HaoChen, Adrien Gaidon, and Tengyu Ma. Self-supervised learning is more robust to dataset imbalance. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2022.

[59] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proc.Int.Conf.on Computer Vision (ICCV)*, 2021.

[60] Linus Ericsson, Henry Gouk, and Timothy M. Hospedales. How well do self-supervised models transfer? In *Proc.Int.Conf.on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[61] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[62] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2017.

[63] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[64] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosiute, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemí Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: Harmlessness from AI feedback. *CoRR*, abs/2212.08073, 2022.

[65] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.

[66] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S. Dhillon. Node feature extraction by self-supervised multi-

scale neighborhood prediction. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2022.

[67] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2023.

[68] Vassilis N. Ioannidis, Xiang Song, Da Zheng, Houyu Zhang, Jun Ma, Yi Xu, Belinda Zeng, Trishul Chilimbi, and George Karypis. Efficient and effective training of language and graph neural network models. *CoRR*, abs/2206.10781, 2022.

[69] Costas Mavromatis, Vassilis N. Ioannidis, Shen Wang, Da Zheng, Soji Adeshina, Jun Ma, Han Zhao, Christos Faloutsos, and George Karypis. Train your own GNN teacher: Graph-aware distillation on textual graphs. In *Proc.European.Conf.on Machine Learning and Knowledge Discovery in Databases (ECML KDD)*, 2023.

[70] Han Xie, Da Zheng, Jun Ma, Houyu Zhang, Vassilis N Ioannidis, Xiang Song, Qing Ping, Sheng Wang, Carl Yang, Yi Xu, et al. Graph-aware language model pre-training on a large graph corpus can help multiple graph applications. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5270–5281, 2023.

[71] Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael M. Bronstein, Zhaocheng Zhu, and Jian Tang. Graphtext: Graph reasoning in text space. *CoRR*, abs/2310.01089, 2023.

[72] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. *CoRR*, abs/2310.04560, 2023.

[73] Jiayan Guo, Lun Du, and Hengyu Liu. GPT4Graph: Can large language models understand graph structured data ? An empirical evaluation and benchmarking. *CoRR*, abs/2305.15066, 2023.

[74] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. GraphGPT: Graph instruction tuning for large language models. *CoRR*, abs/2310.13023, 2023.

[75] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. GraphFormers: GNN-nested transformers for representation learning on textual graph. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2021.

[76] Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, Weiwei Deng, Qi Zhang, Lichao Sun, Xing Xie, and Senzhang Wang. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

[77] Yue Liu, Ke Liang, Jun Xia, Sihang Zhou, Xihong Yang, Xinwang Liu, and Stan Z. Li. Dink-net: Neural clustering on large graphs. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2023.

[78] Fnu Devvrit, Aditya Sinha, Inderjit S. Dhillon, and Prateek Jain. S3GC: Scalable self-supervised graph clustering. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2022.

[79] Puja Trivedi, Ekdeep Singh Lubana, Yujun Yan, Yaoqing Yang, and Danai Koutra. Augmentations in graph contrastive learning: Current methodological flaws & towards better practices. In *Proc.Acm Conf.on World Wide Web (WWW)*, 2022.

[80] Puja Trivedi, Ekdeep Singh Lubana, Mark Heimann, Danai Koutra, and Jayaraman J. Thiagarajan. Analyzing data-centric properties for graph contrastive learning. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2022.

[81] Puja Trivedi, Mark Heimann, Rushil Anirudh, Danai Koutra, and Jayaraman J. Thiagarajan. Estimating epistemic uncertainty of graph neural networks. In *Data Centric Machine Learning Workshop @ ICML*, 2023.

[82] Puja Trivedi, Danai Koutra, and Jayaraman J Thiagarajan. On estimating link prediction uncertainty using stochastic centering. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024.

[83] Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M. Bronstein. Fake news detection on social media using geometric deep learning. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[84] Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. Rumor detection on social media with bi-directional graph convolutional networks. In *Proc.Association for the Advancement of Artificial Intelligence Conf.on Artificial Intelligence (AAAI)*, 2020.

[85] Yi Han, Shanika Karunasekera, and Christopher Leckie. Continual learning for fake news detection from social media. In *Int.Conf.on Artificial Neural Networks*, 2021.

[86] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv*, abs/2003.00982, 2020.

[87] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. In *Proc. ACM Comput. Surv.*, 2023.

[88] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Proc. Conf. Neural Information Processing Systems, NeurIPS*, 2020.

[89] Tao Guo and Baojiang Cui. Web page classification based on graph neural network. In *Innovative Mobile and Internet Services in Ubiquitous Computing*, 2022.

[90] Yingtong Dou, Kai Shu, Congying Xia, Philip S. Yu, and Lichao Sun. User preference-aware fake news detection. In *Proc.of the Int.Acm SIGIR Conf.on Research and Development in Information Retrieval*, 2021.

[91] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. Karate club: An API oriented open-source python framework for unsupervised learning on graphs. In *Proc.of ACM Int.Conf.on Information and Knowledge Management (CIKM)*, 2020.

[92] Yujun Yan, Jiong Zhu, Marlena Duda, Eric Solarz, Chandra Sekhar Sripada, and Danai Koutra. GroupINN: Grouping-based interpretable neural network for classification of limited, noisy brain data. In *Proc.Int. Conf. on Knowledge Discovery & Data Mining, KDD*, 2019.

[93] Nikil Wale and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. In *Int.Conf.on Data Mining (ICDM)*, 2006.

[94] Mark Heimann, Tara Safavi, and Danai Koutra. Distribution of node embeddings as multiresolution features for graphs. In *2019 IEEE International Conference on Data Mining (ICDM)*, 2019.

[95] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2020.

[96] Doyeong Hwang, Soojung Yang, Yongchan Kwon, Kyung-Hoon Lee, Grace Lee, Hanseok Jo, Seyeol Yoon, and Seongok Ryu. Comprehensive study on molecular supervised learning with graph neural networks. *Journal of Chemical Information and Modeling*, 60(12):5936–5945, 2020.

[97] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2021.

[98] Yin Fang, Haihong Yang, Xiang Zhuang, Xin Shao, Xiaohui Fan, and Huajun Chen. Knowledge-aware contrastive molecular graph learning. *arXiv*, 2021.

[99] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2020.

[100] Julius von Kügelgen, Yash Sharma, Luigi Gresele, Wieland Brendel, Bernhard Schölkopf, Michel Besserve, and Francesco Locatello. Self-supervised learning with data augmentations provably isolates content from style. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2021.

[101] Roland Zimmerman, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive learning inverts the data generating process. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2021.

[102] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2020.

[103] Zixin Wen and Yuanzhi Li. Towards understanding the feature learning process of self-supervised contrastive learning. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2021.

[104] Ben Poole, Sherjil Ozair, Aaron van den Oord, Alexander A. Alemi, and George Tucker. On variational bounds of mutual information. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2019.

[105] Kento Nozawa and Issei Sato. Understanding negative samples in instance discriminative self-supervised representation learning. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2021.

[106] Jordan T.Ash, Surbhi Goel, Akshay Krishnamurthy, and Dipendra Misra. Investigating the role of negatives in contrastive representation learning. In *Int.Conf.on Artificial Intelligence and Statistics (AISTATS)*, 2022.

[107] Joshua David Robinson, Li Sun, Ke Yu, kayhan Batmanghelich, Stefanie Jegelka, and Suvrit Sra. Can contrastive learning avoid shortcut solutions? In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2021.

[108] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proc.Int.Conference on Computer Vision (ICCV)*, 2021.

[109] Ashraful Islam, Chun-Fu Chen, Rameswar Panda, Leonid Karlinsky, Richard J. Radke, and Rogério Feris. A broad study on the transferability of visual representations with contrastive learning. In *Proc.Int.Conference on Computer Vision (ICCV)*, 2021.

[110] Ramprasaath R. Selvaraju, Karan Desai, Justin Johnson, and Nikhil Naik. CASTing your model: Learning to localize improves self-supervised representations. In *Proc.Int.Conf.on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[111] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and beyond (GRL+ 2020)*, 2020.

[112] Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Gavin Taylor, and Tom Goldstein. FLAG: Adversarial data augmentation for graph neural networks. *CoRR*, 2020.

[113] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver J. Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. In *Proc.Association for the Advancement of Artificial Intelligence Conf.on Artificial Intelligence (AAAI)*, 2020.

[114] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2018.

[115] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Proc.Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[116] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. In *Proc.Ieee Conf.on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[117] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. In *Adv.in Neural Information Processing Systems (NeurIPS)*, 2021.

[118] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2020.

[119] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2019.

[120] Yao-Hung Hubert Tsai, Yue Wu, Ruslan Salakhutdinov, and Louis-Philippe Morency. Self-supervised learning from a multi-view perspective. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2021.

[121] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proc.Int.Conf.on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[122] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2020.

[123] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2019.

[124] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Proc.Euro. Conf. on Computer Vision (ECCV)*, 2020.

[125] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report, Stanford InfoLab / Stanford InfoLab, 1999.

[126] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *In Proceedings of the ICML*, 2002.

[127] Weijing Shi and Ragunathan (Raj) Rajkumar. Point-GNN: Graph neural network for 3D object detection in a point cloud. In *Proc.Ieee Conf.on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[128] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. Bringing your own view: Graph contrastive learning without prefabricated data augmentations. In *Proc.Acm Int.Conf.Web Search and Data Mining (WSDM)*, 2022.

[129] Kaveh Hassani and Amir Hosein Khas Ahmadi. Learning graph augmentations to learn graph representations. *arXiv*, 2022.

[130] Zekarias T. Kefato, Sarunas Girdzijauskas, and Hannes Stärk. Jointly learnable data augmentations for self-supervised gnns. *arXiv*, abs/2108.10420, 2021.

[131] Zekarias T. Kefato and Sarunas Girdzijauskas. Self-supervised graph neural networks without explicit negative sampling. In *Int.Workshop on Self-Supervised Learning for the Web (WWW'21)*, 2021.

[132] Namkyeong Lee, Junseok Lee, and Chanyoung Park. Augmentation-free self-supervised learning on graphs. *arXiv*, abs/2112.02472, 2021.

[133] Hyeonjin Park, Seunghun Lee, Sihyeon Kim, Jinyoung Park, Jisu Jeong, Kyung-Min Kim, Jung-Woo Ha, and Hyunwoo J. Kim. Metropolis-hastings data augmentation for graph neural networks. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2021.

[134] Alex Tamkin, Mike Wu, and Noah D. Goodman. Viewmaker networks: Learning views for unsupervised representation learning. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2021.

[135] Tete Xiao, Xiaolong Wang, Alexei A. Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2021.

[136] Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In *Proc.ACM Int.Conf.on Knowledge Discovery & Data Mining (SIGKDD)*, 2015.

[137] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. FakeNewsNet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. In *Big Data*, 2020.

[138] Nils M. Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2012.

[139] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alexander J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. In *Proc.Int.Conf.on Intelligent Systems for Molecular Biology*, 2005.

[140] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research (JMLR)*, 2011.

[141] Zhiping Zeng, Anthony KH Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. Comparing stars: On approximating graph edit distance. In *Proc.of the Very Large Data Base (VLDB) Endowment*, 2009.

[142] Vadeem Safronov. Almost free inductive embeddings out-perform trained graph neural networks in graph classification in a range of benchmarks, 2021.

[143] Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. Self-supervised graph-level representation learning with local and global structure. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2021.

[144] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *ArXiv*, abs/2101.11174, 2021.

[145] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc.Annual Meeting of the Assoc. for Computational Linguistics (ACL)*, 2004.

[146] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2015.

[147] Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. Understanding attention and generalization in graph neural networks. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2019.

[148] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stephane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2021.

[149] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proc.Acm Conf.on World Wide Web (WWW)*, 2020.

[150] Colin Wei, Kendrick Shen, Yining Chen, and Tengyu Ma. Theoretical analysis of self-training with deep networks on unlabeled data. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2021.

[151] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2022.

[152] Weiran Huang, Mingyang Yi, and Xuyang Zhao. Towards the generalization of contrastive self-supervised learning. *arXiv*, abs/2111.00743, 2021.

[153] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2021.

[154] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning, 2021.

[155] Ashwini Pokle, Jinjin Tian, Yuchen Li, and Andrej Risteski. Contrasting the landscape of contrastive and non-contrastive learning. *arXiv preprint arXiv:2203.15702*, 2022.

[156] Liu Ziyin, Ekdeep Singh Lubana, Masahito Ueda, and Hidenori Tanaka. What shapes the loss landscape of self-supervised learning? *arXiv preprint arXiv:2210.00638*, 2022.

[157] Maria-Florina Balcan, Avrim Blum, and Ke Yang. Co-training and expansion: Towards bridging theory and practice. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2004.

[158] Ekdeep Singh Lubana, Chi Ian Tang, Fahim Kawsar, Robert P Dick, and Akhil Mathur. Orchestra: Unsupervised federated learning via globally consistent clustering. *arXiv preprint arXiv:2205.11506*, 2022.

[159] Nikunj Saunshi, Jordan Ash, Surbhi Goel, Dipendra Misra, Cyril Zhang, Sanjeev Arora, Sham Kakade, and Akshay Krishnamurthy. Understanding contrastive learning requires incorporating inductive biases. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2022.

[160] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. In *Bayesian Deep Learning Workshop (NeurIPS)*, 2016.

[161] William Falcon, Ananya Harsh Jha, Teddy Koker, and Kyunghyun Cho. AAVAE: Augmentation-augmented variational autoencoders. *CoRR*, 2021.

[162] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2017.

[163] Thomas Gaudelet, Ben Day, Arian R. Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy B. R. Hayter, Richard Vickers, Charles Roberts, Jian Tang, David Roblin, Tom L. Blundell, Michael M. Bronstein, and Jake P. Taylor-King. Utilising graph machine learning within drug discovery and development. *CoRR*, abs/2012.05716, 2020.

[164] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph R-CNN for scene graph generation. In *Proc.Euro. Conf. on Computer Vision (ECCV)*, 2018.

[165] Yanqiao Zhu, Yuanqi Du, Yinkai Wang, Yichen Xu, Jieyu Zhang, Qiang Liu, and Shu Wu. A survey on deep graph generation: Methods and applications. In *Proc.Conf.on Learning on Graphs (LOG)*, 2022.

[166] Dan Hendrycks, Andy Zou, Mantas Mazeika, Leonard Tang, Bo Li, Dawn Song, and Jacob Steinhardt. PixMix: Dreamlike pictures comprehensively improve safety measures. In *Proc.Int.Conf.on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[167] Puja Trivedi, Danai Koutra, and Jayaraman J. Thiagarajan. A closer look at model adaptation using feature distortion and simplicity bias. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2023.

[168] Olivia Wiles, Sven Gowal, Florian Stimberg, Sylvestre-Alvise Rebuffi, Ira Ktena, Krishnamurthy Dj Dvijotham, and Ali Taylan Cemgil. A fine-grained analysis on distribution shift. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2022.

[169] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Proc.Adv.in Neural Information Processing Systems NeurIPS*, 2019.

[170] Kartik Gupta, Amir Rahimi, Thalaiyasingam Ajanthan, Thomas Mensink, Cristian Sminchisescu, and Richard Hartley. Calibration of neural networks using splines. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2021.

[171] Meelis Kull, Miquel Perelló-Nieto, Markus Kängsepp, Telmo de Menezes e Silva Filho, Hao Song, and Peter A. Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with Dirichlet calibration. In *Proc.Adv.in Neural Information Processing Systems NeurIPS*, 2019.

[172] Jize Zhang, Bhavya Kailkhura, and Thomas Yong-Jin Han. Mix-n-match : Ensemble and compositional methods for uncertainty calibration in deep learning. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2020.

[173] Jayaraman J. Thiagarajan, Rushil Anirudh, Vivek Narayanaswamy, and Peer-Timo Bremer. Single model uncertainty estimation via stochastic data centering. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2022.

[174] Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. Deep anomaly detection with outlier exposure. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2019.

[175] José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2015.

[176] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2015.

[177] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Üstebay. Bayesian graph convolutional neural networks for semi-supervised classification. In *AAAI*, 2019.

[178] Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. Bayesian graph neural networks with adaptive connection sampling. In *ICML*, 2020.

[179] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2017.

[180] Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Proc.Adv.in Neural Information Processing Systems NeurIPS*, 2020.

[181] Rushil Anirudh and Jayaraman J. Thiagarajan. Out of distribution detection via neural network anchoring. In *Asian Conference on Machine Learning, ACML 2022, 12-14 December 2022, Hyderabad, India*, 2022.

[182] Aviv Netanyahu, Abhishek Gupta, Max Simchowitz, Kaiqing Zhang, and Pulkit Agrawal. Learning to extrapolate: A transductive approach. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2023.

[183] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3), 2021.

[184] Mrinank Sharma, Sebastian Farquhar, Eric Nalisnick, and Tom Rainforth. Do bayesian neural networks need to be fully stochastic? In *AISTATS*, 2023.

[185] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2022.

[186] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2022.

[187] Dan Hendrycks, Steven Basart, Mantas Mazeika, Andy Zou, Joseph Kwon, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. Scaling out-of-distribution detection for real-world settings. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2022.

[188] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Proc.Adv.in Neural Information Processing Systems NeurIPS*, 2018.

[189] Haoqi Wang, Zhizhong Li, Litong Feng, and Wayne Zhang. ViM: Out-of-distribution with virtual-logit matching. In *Proc.Int.Conf.on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[190] Weitang Liu, Xiaoyun Wang, John D. Owens, and Yixuan Li. Energy-based out-of-distribution detection. In *Proc.Adv.in Neural Information Processing Systems NeurIPS*, 2020.

[191] Konstantin Kirchheim, Marco Filax, and Frank Ortmeier. PyTorch-OOD: A library for out-of-distribution detection based on PyTorch. In *Workshop at the Proc.Int.Conf.on Computer Vision and Pattern Recognition CVPR*, 2022.

[192] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2021.

[193] Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampásek. Attending to graph transformers. *CoRR*, abs/2302.04181, 2023.

[194] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2021.

[195] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *Proc.Int.Conf.on Learning Representations ICLR*, 2022.

[196] Vijay Prakash Dwivedi, Ladislav Rampásek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Proc.Adv.in Neural Information Processing Systems NeurIPS, Datasets and Benchmark Track*, 2022.

[197] Yayong Li, Jie Yin, and Ling Chen. SEAL: Semisupervised adversarial active learning on attributed graphs. *IEEE Trans. Neural Networks Learn. Syst.*, 32(7):3136–3147, 2021.

[198] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2016.

[199] Glenn W. Brier. Verification of forecasts expressed in terms of probability. In *Monthly Weather Review*, 1950.

[200] Jian Kang, Qinghai Zhou, and Hanghang Tong. JuryGCN: Quantifying jackknife uncertainty on graph convolutional networks. In *Proc.Int. Conf. on Knowledge Discovery & Data Mining, KDD*, 2022.

[201] Felix L. Opolka and Pietro Liò. Bayesian link prediction with deep graph convolutional gaussian processes. In *International Conference on Artificial Intelligence and Statistics,AISTATS*, 2022.

[202] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[203] M. E. J. Newman and Gesine Reinert. Estimating the number of communities in a network. *CoRR*, abs/1605.02753, 2016.

[204] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012*, pages 745–754. IEEE Computer Society, 2012.

[205] Xiaoxin He, Xavier Bresson, Thomas Laurent, and Bryan Hooi. Explanations as features: LLM-based features for text-attributed graphs. *CoRR*, abs/2305.19523, 2023.

[206] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. Efficient graph convolution for joint node representation learning and clustering. In K. Selcuk Candan, Huan Liu, Leman Akoglu, Xin Luna Dong, and Jiliang Tang, editors, *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*, pages 289–297. ACM, 2022.

[207] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2020.

[208] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2018.

[209] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *Journal of Machine Learning Research*, 2023.

[210] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *CoRR*, abs/2009.00236, 2020.

[211] Jiaqi Ma, Ziqiao Ma, Joyce Chai, and Qiaozhu Mei. Partition-based active learning for graph neural networks. 2023, 2023.

[212] Seyed Mehran Kazemi, Anton Tsitsulin, Hossein Esfandiari, MohammadHossein Bateni, Deepak Ramachandran, Bryan Perozzi, and Vahab S. Mirrokni. Tackling provably hard representative selection via graph neural networks. *CoRR*, abs/2205.10403, 2022.

[213] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *CoRR*, abs/2312.02783, 2023.

[214] Hongyuan Zhang, Pei Li, Rui Zhang, and Xuelong Li. Embedding graph auto-encoder for graph clustering. *IEEE Trans. Neural Networks Learn. Syst.*, 2023.

[215] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. Structural deep clustering network. In *Proc.Int.Web Conf.(WebConf)*, 2020.

[216] Lei Gong, Sihang Zhou, Wenxuan Tu, and Xinwang Liu. Attributed graph clustering with dual redundancy reduction. In *Proc. of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI*, 2022.

[217] Jun Xia, Lirong Wu, Ge Wang, Jintao Chen, and Stan Z. Li. ProGCL: Rethinking hard negative mining in graph contrastive learning. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2022.

[218] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. StructGPT: A general framework for large language model to reason over structured data. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore, December 2023. Association for Computational Linguistics.

[219] Hao Liu, Jiarui Fend, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *The Twelfth International Conference on Learning Representations*, 2024.

[220] Peter West, Chandra Bhagavatula, Jack Hessel, Jena D Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic knowledge distillation: From general language models to commonsense models. *arXiv preprint arXiv:2110.07178*, 2021.

[221] Jordan T. Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[222] Dan Wang and Yi Shang. A new active labeling method for deep learning. In *2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6-11, 2014*, pages 112–119. IEEE, 2014.

[223] Melanie Ducoffe and Frédéric Precioso. Adversarial active learning for deep networks: A margin based approach. *CoRR*, abs/1802.09841, 2018.

[224] Juncheng Liu, Yiwei Wang, Bryan Hooi, Renchi Yang, and Xiaokui Xiao. Active learning for node classification: The additional learning ability from unlabelled nodes. *CoRR*, abs/2012.07065, 2020.

[225] Natalia Ostapuk, Jie Yang, and Philippe Cudré-Mauroux. ActiveLink: Deep active learning for link prediction in knowledge graphs. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, 2019.

[226] Yuwei Zhang, Zihan Wang, and Jingbo Shang. ClusterLLM: Large language models as a guide for text clustering. In *Proc.Int.Conf.on Empirical Methods in Natural Language Processing, (EMNLP)*, 2023.

[227] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[228] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2020.

[229] Vijay Viswanathan, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. Large language models enable few-shot clustering. *CoRR*, abs/2307.00524, 2023.

[230] Chau Minh Pham, Alexander Miserlis Hoyle, Simeng Sun, and Mohit Iyyer. TopicGPT: A prompt-based topic modeling framework. *CoRR*, abs/2311.01449, 2023.

[231] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*. Association for Computational Linguistics, 2023.

[232] Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, 2023.

[233] Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine A. Heller, and Subhrajit Roy. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. 2023.

[234] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[235] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[236] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[237] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[238] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[239] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

[240] Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *arXiv preprint arXiv:2201.12122*, 2022.

[241] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[242] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[243] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.

[244] Lichao Sun, Yingtong Dou, Carl Yang, Kai Zhang, Ji Wang, Philip S. Yu, Lifang He, and Bo Li. Adversarial attack and defense on graph data: A survey. *IEEE Trans. Knowl. Data Eng.*, 35(8):7693–7711, 2023.

[245] Arijit Ghosh Chowdhury, Md Mofijul Islam, Vaibhav Kumar, Faysal Hossain Shezan, Vaibhav Kumar, Vinija Jain, and Aman Chadha. Breaking down the defenses: A comparative survey of attacks on large language models. *CoRR*, abs/2403.04786, 2024.

[246] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S. Dhillon. Node feature extraction by self-supervised multi-scale neighborhood prediction. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

[247] Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. Simteg: A frustratingly simple approach improves textual graph learning. 2023.

[248] Yichuan Li, Kaize Ding, and Kyumin Lee. GRENADE: graph-centric language model for self-supervised representation learning on text-attributed graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, 2023.

[249] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[250] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. 2019.

[251] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.*, 24(5):513–523, 1988.

[252] Yige Li, Hanxun Huang, Yunhan Zhao, Xingjun Ma, and Jun Sun. Backdoorllm: A comprehensive benchmark for backdoor attacks on large language models. 2024.

[253] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In *Proc. Conf. Neural Information Processing Systems, NeurIPS*, 2023.

[254] Fredrik Nestaas, Edoardo Debenedetti, and Florian Tramèr. Adversarial search engine optimization for large language models. *CoRR*, abs/2406.18382, 2024.

[255] Harsh Chaudhari, Giorgio Severi, John Abascal, Matthew Jagielski, Christopher A. Choquette-Choo, Milad Nasr, Cristina Nita-Rotaru, and Alina Oprea. Phantom: General trigger attacks on retrieval augmented language generation. *CoRR*, abs/2405.20485, 2024.

[256] Sahar Abdelnabi, Kai Greshake, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, AISec 2023, Copenhagen, Denmark, 30 November 2023*, pages 79–90. ACM, 2023.

[257] Aounon Kumar and Himabindu Lakkaraju. Manipulating large language models to increase product visibility. *CoRR*, abs/2404.07981, 2024.

[258] Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of what art? A call for multi-prompt LLM evaluation. *Trans. Assoc. Comput. Linguistics*, 2024.

[259] Jens Hauser, Zhao Meng, Damián Pascual, and Roger Wattenhofer. BERT is robust! A case against synonym-based adversarial examples in text classification. *CoRR*, abs/2109.07403, 2021.

[260] Steffen Eger, Gözde Gül Sahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. Text processing like humans do: Visually attacking and shielding NLP systems. In *Proc. Conf. of the North American Chapter ofthe Assoc. for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Association for Computational Linguistics, 2019.

[261] Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2024.

[262] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and k Dejing Dou. Hotflip: White-box adversarial examples for text classification. In *Proc. Assoc. for Computational Linguistics (ACL-ShortPapers)*. Association for Computational Linguistics, 2018.

[263] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *Proc. Conf. on Artificial Intelligence (AAAI)*, 2020.

[264] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. BERT-ATTACK: adversarial attack against BERT using BERT. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[265] Jiong Zhu, Junchen Jin, Donald Loveland, Michael T. Schaub, and Danai Koutra. How does heterophily impact the robustness of graph neural networks?: Theoretical connections and practical implications. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, 2022.

[266] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. In *Neural Information Processing Systems, NeurIPS*, 2021.

[267] Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. Can llms effectively leverage graph structural information: When and why, 2023.

[268] Dong Lu, Tianyu Pang, Chao Du, Qian Liu, Xianjun Yang, and Min Lin. Test-time backdoor attacks on multimodal large language models. 2024.

[269] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[270] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[271] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. Scikit-image: Image processing in Python, 2014.

[272] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. GCC: Graph contrastive coding for graph neural network pre-training. In *Proc.ACM Int.Conf.on Knowledge Discovery & Data Mining (SIGKDD)*, 2020.

[273] Yixin Liu, Shirui Pan, Ming Jin, Chuan Zhou, Feng Xia, and Philip S. Yu. Graph self-supervised learning: A survey. *IEEE Trans.on Knowledge and Data Engineering*, 2022.

[274] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative image inpainting with contextual attention. In *Proc.Int.Conf.on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[275] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. When does self-supervision help graph convolutional networks? In *Proc.Int.Conf.on Machine Learning (ICML)*, 2020.

[276] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2019.

[277] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John P. Dickerson, Christoph Studer, Larry S. Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Proc.Adv.in Neural Information Processing Systems (NeurIPS)*, 2019.

[278] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 694–699, 2002.

[279] Amir Rahimi, Amirreza Shaban, Ching-An Cheng, Richard Hartley, and Byron Boots. Intra order-preserving functions for calibration of multi-class neural networks. *Advances in Neural Information Processing Systems*, 33:13456–13467, 2020.

[280] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proc.Conf.on Adv.of Artificial Intelligence (AAAI)*, 2015.

[281] Ananya Kumar, Percy Liang, and Tengyu Ma. Verified uncertainty calibration. In *Proc.Adv.in Neural Information Processing Systems NeurIPS*, 2019.

[282] Nicki Skafte Detlefsen, Jiri Borovec, Justus Schock, Ananya Harsh, Teddy Koker, Luca Di Liello, Daniel Stancl, Changsheng Quan, Maxim Grechkin, and William Falcon. TorchMetrics - measuring reproducibility in PyTorch, 2022.

[283] Saurabh Garg, Sivaraman Balakrishnan, Zachary C. Lipton, Behnam Neyshabur, and Hanie Sedghi. Leveraging unlabeled data to predict out-of-distribution performance. In *Proc.Int.Conf.on Learning Representations (ICLR)*, 2022.

[284] Nathan Ng, Neha Hulkund, Kyunghyun Cho, and Marzyeh Ghassemi. Predicting out-of-domain generalization with local manifold smoothness. *CoRR*, abs/2207.02093, 2022.

[285] Puja Trivedi, Danai Koutra, and Jayaraman J Thiagarajan. A closer look at scoring functions and generalization prediction. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023.

[286] Devin Guillory, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. Predicting with confidence on unseen distributions. In *ICCV*, 2021.

[287] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *Proc.Int.Conf.on Computer Vision and Pattern Recognition (CVPR)*, 1997.

[288] Michael D Plummer and László Lovász. *Matching Theory*. Elsevier, 1986.

[289] Ka Yee Yeung and Walter L Ruzzo. Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data. *Bioinformatics (Oxford, England)*, 17(9):763–774, 2001.

[290] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking. *CoRR*, abs/1707.03815, 2017.

[291] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proc.Int.Conf.on Machine Learning (ICML)*, 2016.

[292] Péter Mernyei and Catalina Cangea. Wiki-CS: A wikipedia-based benchmark for graph neural networks. *CoRR*, abs/2007.02901, 2020.