Understanding Self-Supervised Graph Representation Learning from a Data-Centric Perspective

Puja Trivedi
University of Michigan
pujat@umich.edu

Ekdeep Singh Lubana
University of Michigan
eslubana@umich.edu

Mark Heimann
Lawrence Livermore Natl. Laboratory
heimann2@llnl.gov

Danai Koutra
University of Michigan
dkoutra@umich.edu

Jayaraman J. Thiagarajan
Lawrence Livermore Natl. Laboratory
jjayaram@llnl.gov

ABSTRACT

Recent analyses of self-supervised representation learning (SSL) find the following data-centric assumptions are critical for learning high-quality representations: *invariance* to task-irrelevant semantics, *separability* of classes in some latent space, and *recoverability* of labels from augmented samples. However, it is unlikely that graph SSL methods support these assumptions given the discrete, non-Euclidean nature of graphs. This raises the question: how do graph SSL methods work well? To systematically probe this question, we perform a generalization analysis of graph SSL when using random topological or feature perturbations as generic graph augmentations (GGAs). Our analysis shows how GGAs can effect the recoverability and separability assumptions and theoretically motivates the need for task-relevant data augmentations. Indeed, we empirically find that GGAs fail to induce task-relevant invariances on benchmark datasets, leading to only marginal gains over naive, untrained baselines. Finally, our theory motivates a synthetic data generation process that enables control over both augmentation recoverability and dataset separability., enabling a better benchmark for evaluation of graph SSL methods that demonstrates different training paradigms are effective in different regimes. Overall, our work rigorously contextualizes, both empirically and theoretically, the effects of data-centric assumptions on graph SSL paradigms and augmentations.

1 INTRODUCTION

Self-Supervised Learning (SSL) [1–9] has revolutionized visual representation learning by producing representations that are more robust [10, 11], transferable [12, 13], and semantically consistent [6] than their supervised counterparts. This impressive empirical success has motivated a surge of efforts that seek to gain insights into SSL's behavior [14–21] or adapt successful frameworks to different modalities, including graph data [22–26]. Notably, many analyses of SSL have converged upon the following data-centric assumptions 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 [16, 20, 27] so that representations are semantically consistent for downstream tasks. Due to the continuous representation of natural images and well-designed augmentation strategies, these assumptions are indeed supported by standard visual SSL practices [28].

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 [29–31] 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 towards understanding this relationship by analyzing commonly used generic graph augmentations (GGAs), and designing useful tools that enable probing of these properties, including a theoretical framework for understanding the generalization of graph contrastive learning (CL) with GGA, and a synthetic data generation process that helps disentangle the effects of unrecoverable augmentations from performance. As we show, current practices often do not support the desired properties, and we identify some of the causes for this misalignment. Our contributions are as follows:

- **Analysis of Generalization and Separability.** We provide the first generalization error bound for graph CL when using GGAs. This bound suggests that GGAs induce a performance vs. separability trade-off that is determined by underlying dataset properties. Further, our theory provides a principled justification as to why GGAs are insufficient in many tasks and motivates the need for task-aware augmentations in such situations.
- **Missing Invariance on Benchmark Datasets.** On standard benchmarks, we show that despite improved invariance, models

Correspondence to: Puja Trivedi <pujat@umich.edu>.
This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract No. DE-AC52-07NA27344, Lawrence Livermore National Security, LLC.and was supported by the LLNL-LDRD Program under Project No. 21-ERD-012

trained with GGA have marginal improvements in accuracy compared to untrained encoders. Our analysis indicates GGAs induce limited task-relevant invariance at best, revealing a fundamental misalignment between the objectives of CL and its behavior in practice.

- **Recoverability: Synthetic Data Generation Process.** We propose a synthetic data generation process that allows for control over augmentation recoverability. Using the proposed process, we confirm that adhering to the aforementioned properties is indeed beneficial. Notably, even the recent automated graph augmentation methods fail to identify augmentations that induce task-relevant invariances. For reproducibility, we have included code at https://anonymous.4open.science/r/gcl_recon_22-F44C/

2 BACKGROUND

In this section, we briefly discuss existing graph SSL paradigms. We then discuss the motivation behind the data-centric assumptions (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 positive (negative) views of a sample have high (low) representational similarity. Existing SSL frameworks can be broadly categorized based on the mechanism adopted for enforcing this consistency between views: contrastive learning (CL) frameworks [1, 7, 8, 22, 29, 31], such as GraphCL[22], use the InfoNCE loss to perform instance discrimination; approaches that rely only on positive pairs, such as SimSiam [2] and BGRL [24] use Siamese architectures with stop gradient [2] and asymmetric branches[21] respectively; SpecCL [15] uses a spectral clustering loss to enforce consistency; others attempt to directly reduce redundancy between views [3, 32]. Despite these differences, all methods rely upon data augmentation to generate positive views, which are assumed to share semantics.

Generic graph augmentations (GGAs) [22], which include random node dropping, edge perturbation, masking node attributes and sampling subgraphs, are a popularly used graph augmentation strategy and assume that limited changes to a graph's attributes or topology are unlikely to alter a its label. Other strategies include using diffusion matrices [23], GGAs with a non-uniform prior, or automated methods that utilize a bi-level optimization to learn augmentation sampling strategies [29] or parameterized augmentation modules [31]. We primarily focus on GGAs due to their popularity, simplicity and effectiveness. Reconstruction approaches are an alternative paradigm for graph SSL. For example, AAVAE [33] is a novel auto-encoding framework for visual CL that optimizes for recovering original samples from their augmentations. We adapt AAVAE for graph data by using a graph-specific auto-encoder [34] and augmentation strategy. Please the see appendix for additional discussion about augmentations and paradigms.

Theoretical Analsyis of SSL. Several different perspectives have recently been used to successfully analyze SSL's behavior, including learning theory [14, 15, 35], causality [17, 18], information theory [27], and dynamical systems [36]. Many of these analyses assume, either implicitly [18, 35] or explicitly [15, 28, 37], the existence of a latent space that is *invariant* to augmentation functions and supports the properties of *recoverability* and *separation*.

Invariance to Augmentations: Producing a similar representation for an augmentation function induces invariance to the corresponding transform. Indeed, if augmentations are related by properties that are *not relevant* to the downstream task (e.g., rotation angle), representations will become *invariant* to this relationship over the course of SSL training and generalization will improve [16, 38]. 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) [20, 39]. This latter point is often ignored by the theoretical analyses mentioned above, but we take it into account in the following sections.

Recoverability and Separability: These properties state that in the latent space that 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 [15, 28]. While originally proposed for manifolds [37], both recoverability and separability have been recently converted to graph connectivity properties [15] and verified empirically on modern deep learning methods [28]. 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. 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 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. [15], called SpecLoss. Due to the similarities in generalization analysis of contrastive frameworks proposed by Saunshi et al. [14, 38], the takeaways from our work remain valid for other methods as well. We denote SpecLoss using $\mathcal{L}(f)$ and define it 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 defined as: $\mathcal{L}(f) = -2 \cdot \mathbb{E}_{\overline{\boldsymbol{g}}, \boldsymbol{g}^+}\left[f(\overline{\boldsymbol{g}})^\top f(\boldsymbol{g}^+)\right] + \mathbb{E}_{\overline{\boldsymbol{g}}, \boldsymbol{g}^-}\left[\left(f(\overline{\boldsymbol{g}})^\top f(\boldsymbol{g}^-)\right)^2\right]$.

3 GENERALIZATION BOUNDS FOR CL WITH GGAS

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. In this section, we demonstrate how GGAs influence these assumptions by deriving a generalization bound tailored for graph data. Notably, this bound allows us to demonstrate conditions where using

Table 1: Generic Graph Augmentations vs. Graph Edit Operators. GGA can be straightforwardly expressed using graph edit operators. Please see Appendix. B 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

GGAs results in low separability and recoverability, motivating the need for augmentations that induce task-relevant invariances to go beyond perturbative generic 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 3.1 (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,

$$GED\left(g_1, g_2\right) = \min_{\left(e_1,\ldots,e_k\right)\in\mathcal{P}\left(g_1,g_2\right)} \sum_{i=1}^{k} c\left(e_i\right),$$

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 1, frequently used GGA transforms can be easily decomposed using standard graph edit operators described in Def. 3.1. 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, as follows:

Lemma 3.2. ***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 Appendix B, 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 [15]:

Assumption 3.3 (Separability plus Recoverability Assumption, (Assm. 3.5 in [15]))**.** 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.

Intuitively, the assumption 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. 3.3 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/augmentation strength can influence the error of h. We 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 3.2, we can determine if two augmentations could have been generated from the same sample.

Corollary 3.4. *(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 3.5 (Inconsistent Samples)**.** Let $\boldsymbol{g} \in \mathcal{X}$, and $y : \overline{\mathcal{X}} \rightarrow r$ 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. 3.4) 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*.

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, $\mathrm{S}_h(\boldsymbol{g})$. 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 3.6 (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 $\boldsymbol{g} \in \mathrm{S}_i \wedge \boldsymbol{g'} \in \mathrm{S}_j$ 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 3.7 (Error bound due to Inconsistent Samples)**.** *The error of h can be lower-bounded as*

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

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. [15] 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 3.8 (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

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

Intuitively, we use the partitions induced by h as a proxy for class labels and co-occurrence as a notion of similarity (see Lemma 3.2). 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 [15].

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. 3.4) and dissimilar partitions (Def. 3.8). Notably, we will decompose bound in terms of the number of co-occurring augmentation-pairs within 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 $\mu = \sum_{\boldsymbol{g} \in S_i, \boldsymbol{g'} \notin S_i} \mathbb{1}(GED(\boldsymbol{g}, \boldsymbol{g'}) \leq 2\delta)$.

Theorem 3.9 (Generalization Bound for SpecLoss with GGA)**.** *Assume the representation dimension $k \geq 2r$ and Assm. 3.7 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:*

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

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. 5 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 cross 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 4, we find that GGAs are unable to induce such invariances on benchmark datasets.

Recoverability. As shown in Thm. 3.9, better recoverability will improve the tightness of the generalization bound. However, we see that from Coll. 3.7, 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 A CLOSER LOOK AT THE EFFECTIVENESS OF INVARIANCE TO GGA

In the preceding section, we demonstrated how recoverability and separability bounds generalization error.

Though computing these properties directly is intractable on benchmark datasets, our analysis above graph-based learning and

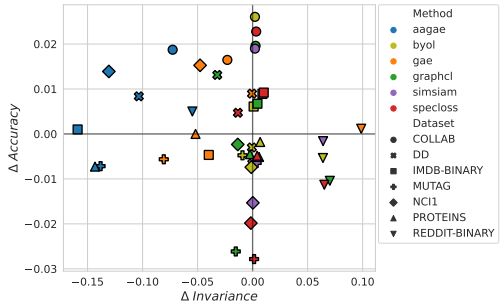

Figure 1: Invariance vs. KNN Acc. The change in invariance (Inv.) and accuracy w.r.t. to an untrained model is plotted, where Inv. is measured according to [19]. Noticeably, Inv. has not significantly increased for many datasets/methods, improved Inv. does not necessarily entail better performance (see Reddit), and AAGAE/GAE often sees decreased Inv., which we suspect is due to using a decoder.

prior works on visual learning [17, 18, 20] 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 augmentation techniques 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 indeed the case, our analysis suggests we should see improvement in performance with increased invariance. To this end, we perform the following experiment.

Experimental Setup: We evaluate seven graph SSL methods on seven, popular benchmark datasets. Specifically, we use the following representative algorithms: (i) *GraphCL* [22], a popular and effective graph CL method; (ii) *GAE*, Graph Autoencoder [34] that uses a reconstruction cost to learn representations; (iii) Augmentation-Augmented Autoencoder [33], 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 [15] for contrastive training; (v) *SimSiam* [2], a positive-sample-only framework that uses stop gradient; (vi) *BYOL* [21], 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 [31, 39–41]. 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 [42] to GAE/AAGAE's decoder for better training. For more experimental details, including the performance of all methods, please see appendix D.

Training, invariance, and what standard benchmarks cannot tell us. To measure whether augmentations have induced invariance, we measure recoverability using the representational similarity measures introduced by Wang and Isola [19]. 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. 1 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 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.

5 EVALUATING GRAPH SSL VIA A NEW SYNTHETIC BENCHMARK

Our analysis indicates the role played by recoverability and separability under task-relevant invariances plays a dramatic role on generalization performance. However, given our results that GGAs do not enable any of 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.

5.1 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.

The design of our data generation process is motivated by a recent theoretical work that seeks to understand how CL, data augmentation, and data generation processes are related. Using a latent variable model, Kugelgen et al. [17] show that self-supervised training with data augmentation is able to recover a *style* vs. *content* partition in the latent representation space. 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. The proposed data generation process creates graph samples that can be decomposed into style vs. content and allows for control over this trade-off (see Figure 3). While designing content-aware augmentations for arbitrary graph datasets is a hard problem [39], with oracle knowledge of the content in this dataset, we can evaluate content-aware augmentations (CAA) with

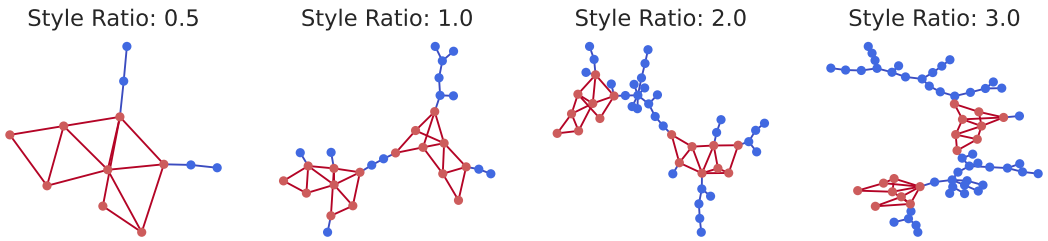

Figure 2: Synthetic Dataset Generation. A class-specific motif (shown in red) completely determines the label, and is therefore considered "content". To vary the amount of style, the size of the background tree graph (shown in blue) is a ratio of the number of "content" nodes. Our dataset goes beyond binary benchmarks and allows for content-aware augmentations, a critical component to understanding graph SSL.

high recoverability at varying levels of separability, approximated 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 [43].

5.2 Identifying Regimes of Success for GGA and CAA

Several real graph datasets can be understood through a style vs. content perspective. For example, in drug discovery tasks [44], 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, 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 [29] and AD-GCL[31]. 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 [1] 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 appendix D for more model details.

Results. Our results in Figure 3 clearly show the value of CAAs. We make several observations. (i) Contrastive methods respond the best to CAAs, yielding high robustness to irrelevant information (style), showing the direct value of incorporating task-relevance. (ii) As the amount of style increases, the problem inherently becomes harder for reconstruction based methods, as the model must learn to reconstruct increasing amounts of irrelevant information. Indeed, we see that the performance of all reconstruction-based methods decreases as style increases. With mild random augmentations, AAGAE performs comparably to GAE, and with aggressive random augmentation it performs worse. Content-aware augmentations significantly improve the performance of AAGAE over baseline augmentations, but it is unable to match that of contrastive methods. (iii) The gain from CAA in high-style regimes is less pronounced for reconstruction approaches than for GCL. This may partially be attributed to increased difficulty in reconstructing larger graphs. It may also suggest that other framework components, such as more expressive architectures [45–49] and sampling strategies [2, 21, 50], must also be developed before reconstruction-based methods are able to see similar success to visual SSL and graph CL. (iv) We see in Figure 3 that automated methods are unable to learn augmentation strategies that induce style invariance. *Indeed, JOAO is unable to find such a solution even when the augmentation prior includes CAAs.* We suspect this is due to their use of bi-level optimization objectives, which are known to be difficult to optimize and prone to finding locally optimal solutions. Overall, this experiment demonstrates that automated methods can be brittle and discovering meaningful content-aware evaluations in the wild is challenging for the current state of the art. This makes the proposed benchmark, where we can define content-aware augmentations with oracle knowledge, valuable to evaluate such methods.

5.3 Invariance vs. Separability

We now use our synthetic benchmark to investigate how invariance balances off with the critical assumption of class separability in the latent space. Invariance, while desirable as discussed previously, if considered in isolation could be trivially satisfied by representation

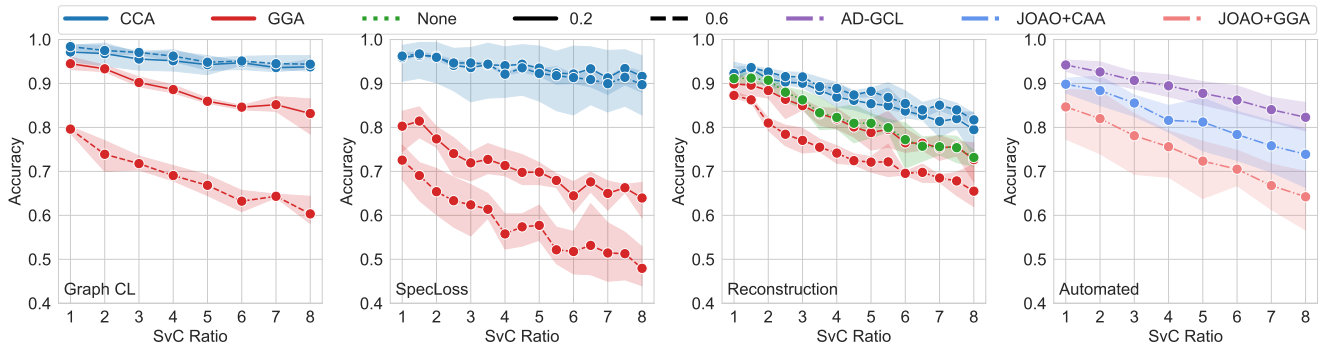

Figure 3: Style Invariance Across Paradigms: We evaluate the performance of contrastive and reconstruction approaches with CAAs and GGAs with varying style vs. content ratios. As expected, reconstruction methods perform best in low style regimes, and CAAs improve graph CL performance. Notably, AD-GCL and JOAO do not learn augmentations that induce style-invariance. JOAO is unable to find such a solution even when the prior augmentation set is expanded to include CAAs.

collapse, where all graphs are mapped to highly similar representations and are not meaningful for distinguishing classes.

Experimental Setup. Using a synthetic dataset at $\kappa = 6$, we train GraphCL with *content-preserving* edge dropping 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 augmented versions. 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. Generic graph augmentations trade off separability for invariance by collapsing representations. Figure 4 shows kernel density estimates of the number of samples that have a given invariance versus separability, using both random and content-preserving augmentations. Representations from the model trained using GGA have somewhat higher invariance but much lower separability scores. This is likely evidence of model collapse; indeed, with a higher augmentation strength of 60%, we found that using GGA produced invariance and separability scores very close to 1 for all samples, indicating that all samples had similar representations (i.e. strong collapse). On the other hand, CAAs help GraphCL achieve over an order of magnitude higher separability and still preserve comparably high invariance. We observed similar trends for SpecLoss.

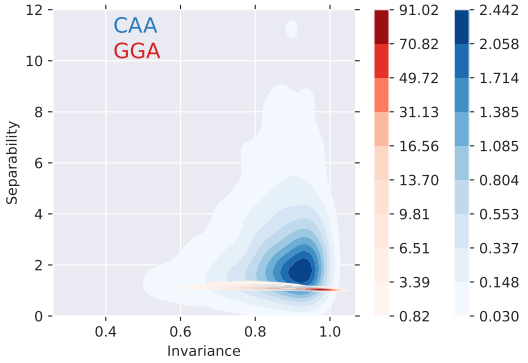

Figure 4: Invariance vs. Separability. On our synthetic data with style-to-content ratio $\kappa = 6$ and 20% augmentation strength, GraphCL trained with random augmentations produces representations with high invariance but low separability, indicating model collapse. In contrast, using content preserving augmentations lead to almost as high invariance but much greater separability.

6 CONCLUSION

In this work, we study the interplay between data-dependent properties, such as recoverability of augmentations and class separability, and the efficacy of self-supervised graph representation learning approaches. We first theoretically obtain the generalization error bound for GCL with popular, generic graph augmentations and show that it is linked to the average graph edit distance between classes. Next, we demonstrate that popular, generic graph augmentations do not induce invariance that is necessarily useful to the downstream tasks. Finally, to better understand the benefits of recoverable, (i.e. content-aware) augmentations, we introduce a new synthetic data generation process, and identify regimes of success for GAA and content-aware augmentations. In summary, our work offers an empirical and analytical framework to develop unsupervised graph representation learning algorithms that are better aligned with data-dependent properties.

REFERENCES

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.
[2] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *CoRR*, abs/2011.10566, 2020.
[3] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.
[4] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735. IEEE, 2020.
[5] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.
[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[7] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *ECCV*, 2020.

[8] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, 2018.

[9] 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 *ICLR*, 2019.

[10] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In *In Proc. of NeurIPS*, 2019.

[11] Hong Liu, Jeff Z. HaoChen, Adrien Gaidon, and Tengyu Ma. Self-supervised learning is more robust to dataset imbalance. *CoRR*, abs/2110.05025, 2021.

[12] Linus Ericsson, Henry Gouk, and Timothy M. Hospedales. How well do self-supervised models transfer? In *CVPR*, 2021.

[13] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *ICLR*, 2020.

[14] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *CoRR*, 2019.

[15] Jeff Z. HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *CoRR*, 2021.

[16] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *NeurIPS*, 2020.

[17] 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. *CoRR*, abs/2106.04619, 2021.

[18] Roland S. Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. Contrastive learning inverts the data generating process. In *ICML*, 2021.

[19] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, 2020.

[20] Senthil Purushwalkam and Abhinav Gupta. Demystifying contrastive self-supervised learning: Invariances, augmentations and dataset biases, 2020.

[21] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *NeurIPS*, 2020.

[22] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *NeurIPS*, 2020.

[23] Kaveh Hassani and Amir Hosein Khas Ahmadi. Contrastive multi-view representation learning on graphs. In *ICML*, 2020.

[24] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Velickovic, and Michal Valko. Bootstrapped representation learning on graphs. *CoRR*, abs/2102.06514, 2021.

[25] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. *WWW*, 2020.

[26] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*, 2020.

[27] Yao-Hung Hubert Tsai, Yue Wu, Ruslan Salakhutdinov, and Louis-Philippe Morency. Self-supervised learning from a multi-view perspective. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[28] Colin Wei, Kendrick Shen, Yining Chen, and Tengyu Ma. Theoretical analysis of self-training with deep networks on unlabeled data. In *ICLR*, 2021.

[29] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *ICML*, 2021.

[30] 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.

[31] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. *NeurIPS*, 2021.

[32] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *CoRR*, 2021.

[33] William Falcon, Ananya Harsh Jha, Teddy Koker, and Kyunghyun Cho. AAVAE: augmentation-augmented variational autoencoders. *CoRR*, 2021.

[34] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *CoRR*, 2016.

[35] Weiran Huang, Mingyang Yi, and Xuyang Zhao. Towards the Generalization of Contrastive Self-Supervised Learning. *arXiv*, abs/2111.00743, 2021.

[36] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised Learning Dynamics without Contrastive Pairs. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2021.

[37] Maria-Florina Balcan, Avrim Blum, and Ke Yang. Co-training and expansion: Towards bridging theory and practice. In *NIPS*, 2004.

[38] Nikunj Saunshi, Jordan Ash, Surbhi Goel, Dipendra Misra, Cyril Zhang, Sanjeev Arora, Sham Kakade, and Akshay Krishnamurthy. Understanding contrastive learning requires incorporating inductive biases, 2022.

[39] Puja Trivedi, Ekdeep Singh Lubana, Yujun Yan, Yaoqing Yang, and Danai Koutra. Augmentations in graph contrastive learning: Current methodological flaws & towards better practices. *CoRR*, abs/2111.03220, 2021.

[40] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[41] Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. Self-supervised graph-level representation learning with local and global structure. In *ICML*, 2021.

[42] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

[43] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*, pages 11975–11986. PMLR, 2021.

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

[45] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS 2020*, 2020.

[46] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[47] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[48] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In *NeurIPS*, 2020.

[49] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[50] Y. Kalantidis, M. Bülent Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus. Hard negative mixing for contrastive learning. In *NeurIPS*, 2020.

[51] Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In *SIGKDD*, 2015.

[52] Nils M. Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. In *ICML*, 2012.

[53] 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 *Proceedings Thirteenth International Conference on Intelligent Systems for Molecular Biology*, 2005.

[54] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.*, 2011.

[55] Nikil Wale and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. In *(ICDM*, 2006.

[56] 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 *SIGKDD*. ACM, 2020.

[57] Zekarias T. Kefato and Sarunas Girdzijauskas. Self-supervised graph neural networks without explicit negative sampling. *CoRR*, 2021.

[58] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver J. Woodford, Meng Jiang, and Neil Shah. Data augmentation for graph neural networks. *CoRR*, 2020.

[59] 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 *NeurIPS*, 2019.

[60] Yixin Liu, Shirui Pan, Ming Jin, Chuan Zhou, Feng Xia, and Philip S. Yu. Graph self-supervised learning: A survey. *CoRR*, abs/2103.00111, 2021.

[61] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Generative image inpainting with contextual attention. *CoRR*, abs/1801.07892, 2018.

[62] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. When does self-supervision help graph convolutional networks? *CoRR*, abs/2006.09136, 2020.

[63] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

A REPRODUCIBILITY

For reproducibility, we have included code at https://anonymous.4open.science/r/gcl_recon_22-F44C/. Code is under-development and will be finalized soon.

B GENERIC GRAPH AUGMENTATIONS AND GRAPH EDIT DISTANCE

The key insight for our analysis in Sec. 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. 3. We also discuss how our analysis can be interpreted with respect to the population augmentation graph (PAG) proposed by HaoChen et al. [15].

Table 2: Notation

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 $\mathcal{A}$.

B.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 B.1 (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, $GED\ (g_1, g_2) = \min_{(e_1,\ldots,e_k)\in\mathcal{P}(g_1,g_2)} \sum_{i=1}^{k} c\ (e_i)$, where $\mathcal{P}\ (g_1, g_2)$ 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. 1, elementary graph edit operators can be used to straightforwardly represent the *node dropping, edge perturbation and sub-graph sampling* generic graph augmentations [22]. 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 3: 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 B.2. ***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. 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, $GED(\overline{\boldsymbol{g}}, \boldsymbol{g}) \leq \delta$. □

Lemma B.3. ***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., $\mathcal{A}(\boldsymbol{g}|\overline{\boldsymbol{g}}) = \frac{1}{|\mathcal{A}|}$.

C DETAILS FOR GENERALIZATION ANALYSIS

As discussed in Sec. 2, recent analyses have found that SSL generalization error can be bounded under the assumptions of invariance to relevant augmentations, recoverability, and separability. Indeed, in Sec. 3, we demonstrated how GGAs and GED influence these properties by deriving a generalization bound 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 SpecCL bound. We then performed some additional manipulation to derive the final expression presented in 3.9. Here, we provide the details and proofs behind these steps. We begin by restating the Separability plus Recoverability assumption.

Assumption C.1 (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. 3, that h will incur irreducible error on inconsistent samples, which are defined as follows:

Corollary C.2. ***(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 $\mathcal{A}(\overline{\boldsymbol{g}})$. □

Definition C.3 (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, $\mathrm{S}_h(\boldsymbol{g})$. 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 C.4 (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 $\boldsymbol{g} \in \mathrm{S}_i \wedge \boldsymbol{g}' \in \mathrm{S}_j$ 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, $\boldsymbol{g}' \in \mathrm{S}_j \Leftrightarrow h(\boldsymbol{g}) = j$, so $j \in |Y^*_{in}|$. However, $GED(\overline{\boldsymbol{g}}_1, \boldsymbol{g}) \leq \delta$, so by Lemma 3.2 and Defn. 3.5, $y(\overline{\boldsymbol{g}}_1) = i \in Y^*_{in}$. 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 C.5 (**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 C.6 (**Partition Dissimilarity**, Reproduction of Defn. 3.8)**.** 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

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

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. 3.4) and dissimilar partitions (Def. 3.8). Notably, we decompose bound in terms of the number of co-occurring augmentation-pairs within 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 C.7 (**Generalization Bound for SpecLoss with GGA**, Reproduction of Thm 3.9)**.** *Assume the representation dimension $k \geq 2r$ and Assm. 3.7 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:*

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

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.

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

$$
\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]} \\
&\quad + \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) \\
&\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)
\end{aligned} \tag{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 3.9.

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

Recall, that inconsistent samples can be determined through graph edit distance (Defn. 3.5) between augmented samples. Moreover, that the maximum allowable edit distance between augmented samples is determined by augmentation strength. □

C.1 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 C.8 (Population Augmentation Graph [15])**.** 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 $\overline{\boldsymbol{g}} \sim \mathcal{P}_{\overline{\mathcal{X}}}$:

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

Table 4: *Dataset Description*

Name	Graphs	Classes	Avg. Nodes	Avg. Edges	Domain
IMDB-BINARY [51]	1000	2	19.77	96.53	Social
REDDIT-BINARY [51]	2000	2	429.63	497.75	Social
MUTAG [52]	188	2	17.93	19.79	Molecule
PROTEINS [53]	1113	2	39.06	72.82	Bioinf.
DD [54]	1178	2	284.32	715.66	Bioinf.
NCI1 [55]	4110	2	29.87	32.30	Molecule

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 C.9. ***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 3.4, $w_{\boldsymbol{g}\boldsymbol{g'}} > 0 \Leftrightarrow \mathcal{A}(\boldsymbol{g}|\overline{\boldsymbol{g}}) > 0 \wedge \mathcal{A}(\boldsymbol{g'}|\overline{\boldsymbol{g}}) > 0$. 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 C.10 (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.

D DATASET GENERATION AND EXPERIMENTAL DETAILS

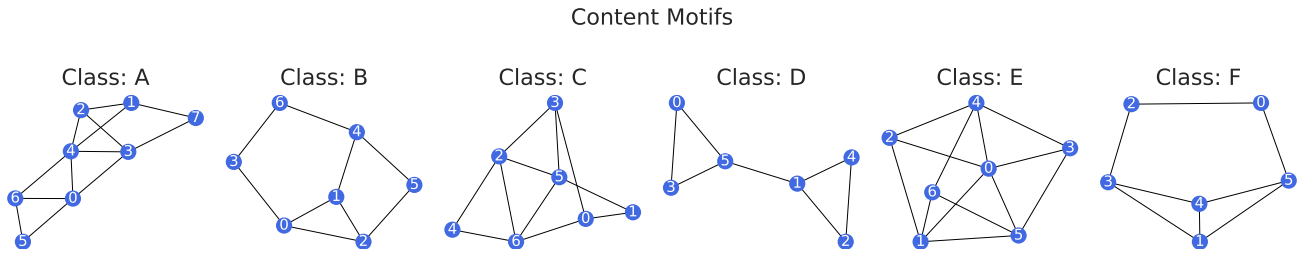

Figure 5: Motifs used to determine class labels.

We use the motifs shown in Fig. D 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 ([46]). 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 [3] 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 5.

Experimental Set-up: We follow You et al. [22] for TUDataset experiments. 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.

E RELATED WORK

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

[3] https://networkx.org/documentation/stable/

Table 5: 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 ([22])	Node Dropping, Edge Adding/Dropping, Attribute Masking, Subgraph Extraction
GCC ([56])	RWR Subgraph Extraction of Ego Network
MVGRL ([23])	PPR Diffusion + Sampling
GCA ([25])	Edge Dropping, Attribute Masking (both weighted by centrality)
BGRL ([24])	Edge Dropping, Attribute Masking
SelfGNN ([57])	Attribute Splitting, Attribute Standardization + Scaling, Local Degree Profile, Paste + Local Degree Profile

properties of a molecular graph. Therefore, only a few works consider graph data augmentation. [58] 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. [30] 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 [59]. 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 [60] for an up-to-date survey. Graph pre-text tasks are often reminiscent of image in-painting tasks [61], and seek to complete masked graphs and/or node features ([13, 62]). Other successful approaches include predicting auxiliary properties of nodes or entire graphs during pre-training or part of regular training to prevent overfitting ([13]). 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. [26, 63] draw inspiration from [9] and maximize the mutual information between global and local representations. MVGRL ([23]) contrasts different views at multiple granularities similar to [8]. [22, 24, 25, 56, 57] use augmentations (which we summarize in Table E) 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.