INTERROGATING PARADIGMS IN SELF-SUPERVISED GRAPH REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph contrastive learning (GCL) is a newly popular paradigm for self-supervised graph representation learning and offers an alternative to reconstruction-based methods. However, it is not well understood what conditions a task must satisfy such that a given paradigm is better suited. In this paper, we investigate the role of dataset properties and augmentation strategies on the success of GCL and reconstruction-based approaches. Using the recent population augmentation graph-based analysis of self-supervised learning, we show theoretically that the success of GCL with popular augmentations is bounded by the graph edit distance between different classes. Next, we introduce a synthetic data generation process that systematically controls the amount of style vs. content in each sample- i.e. information that is irrelevant vs. relevant to the downstream task- to elucidate how graph representation learning methods perform under different dataset conditions. We empirically show that reconstruction approaches perform better when the style vs. content ratio is low and GCL with popular augmentations benefits from moderate style. Our results provide a general, systematic framework for analyzing different graph representation learning methods and demonstrate when a given approach is expected to perform well.

1 APPENDIX

1.1 UNDERSTANDING GENERIC GRAPH AUGMENTATIONS

We expand our discussion on the connections between generic graph augmentations and graph edit distance. We also discuss how the graph edit distance between dataset samples influences the structure of the population augmentation graph HaoChen et al. (2021), a recently introduced tool to understand contrastive learning.

Table 1: Notation

Symbol	Definition
$\overline{\mathcal{X}}$	the original or natural dataset.
$\mathcal{X}$	set of all augmented data.
x_i	data sample containing graph and node feature tuple, (G_i, F_i)
$\mathcal{E}_i$	Edge set of G_i.
$\mathcal{V}_i$	Node set of G_i.
$\gamma \in (0, 1)$	augmentation strength. Controls the $\%$ of edges or nodes that may be perturbed by the selected augmentation
$\mathcal{A}(\overline{x})$	Augmentation, $\mathcal{A}$, applied to the natural sample $\overline{x}$
$\mathcal{A}(\cdot\|\overline{\boldsymbol{x}}_i)$	distribution of augmentations given a natural sample, $\overline{x}_i$.
$\mathcal{B}(\overline{x})$	set of allowable augmentations given $\overline{x}$.
$\mathbf{D} \in \mathbb{Z}^{\|\overline{\mathcal{X}}\times\|\overline{\mathcal{X}}\|}$	positive, symmetric distance matrix where $\mathbf{D}_{i,j} = GED(\overline{x_i}, \overline{x_j})$
$\Gamma \in \mathbb{Z}^{1\times\|\overline{\mathcal{X}}\|}$	column vector containing max. number of allowable perturbations per sample.
$\mathbf{y} \in [0, 1]^{\|\overline{\mathcal{X}}\|}$	column vector providing the labels of all natural samples.

1.1.1 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. Given two graphs, g_1, g_2,

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

where $\mathcal{P}\left(g_1, g_2\right)$ is the set of paths (series of edit operations) that transforms g_1 into g_2, e_i is i-th edit operation in the path, and $c(e_i) > 0$ is the cost of the particular edit. In this work, we consider *node insertion*, *node deletion*, *edge deletion* and *edge addition* as the elementary graph edit operators as these are well-aligned to the augmentations defined in You et al. (2020a), namely *node dropping*, *edge perturbation*, *attribute masking*, and *sub-graph sampling*. While GED is typically defined on graph structure, our analysis can be extended to include categorical node attributes by introducing a graph operator that performs a "replacement" whenever a graph's node attributes disagree. Then, the GED is the cost of structural changes and the number of disagreements between their attributes. Categorical variables are common in molecular classification tasks, where attributes correspond to elements, and discrete node attributes are often used when analyzing GNNs (Xu et al., 2019). We also consider a constant cost of 1 per operation, such that GED counts the number of operations required to transform one graph into another.

For example, let (g, g_a) represent the original and augmented graph respectively, where we perform *node dropping* to obtain g_a. Recall that the *node dropping* augmentation may only drop up to some fraction of nodes in g. Then, clearly the minimum cost path can then be found using only *node deletion* operators, and the $GED(g, g_a)$ is bounded by the number of allowed node drops. Similarly, if g_a was obtained through the *edge perturbation* augmentation, which randomly adds or removes a fraction of edges, then $GED(g, g_a)$ 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, $GED(g, g_a)$ is bounded by $|g| - |g_a|$.

Given the aforementioned discussion, we can now define the set of allowable augmentations using *GED* and make the following remarks. Please see Table 1 for a complete list of notation.

Definition 1 (Set of Allowable Augmentations)**.** *Let $\mathcal{A}$ be a generic graph augmentation (node dropping, etc). Then, all allowable augmented samples induced by $\mathcal{A}(x_i)$ have graph edit distance less than $\max\{\gamma|\mathcal{V}_i|, \gamma|\mathcal{E}_i|\}$ to x_i. Equivalently:*

$$\mathcal{B}(x_i) \triangleq \{x' : GED(x', x_i) \leq \max\{\gamma|\mathcal{V}_i|, \gamma|\mathcal{E}_i|\}\}.$$

Remark 1.1 (Upper-bound on Size of Augmentation Set)**.** *The size of $B(\overline{x_i})$ can be upper-bounded through a combinatorial or counting process. For example, to determine $B(\overline{x_i})$ when the considered augmentation is node dropping, we can delineate all sets of possible nodes with size upto $\gamma|\mathcal{V}_i|$. Formally, the upper-bound on the number of samples generated using node dropping are:*

$$|\mathcal{B}(\overline{x_i})| \leq \sum_{j=1}^{\gamma\mathcal{V}_i} \frac{|\mathcal{V}_i|!}{(|\mathcal{V}_i| - j)!j!}$$

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.

Definition 2 (Overlapping Sample)**.** *An augmented sample, x', is considered an overlapping sample if belong to the augmentation set of multiple natural samples: $x' \in \mathcal{B}(\overline{x}_i) \wedge x' \in \mathcal{B}(\overline{x}_j)$, where $i \neq j$.*

Using Def. 2, we show that overlapping examples must exist given certain conditions on graph edit distance of samples in $\overline{\mathcal{X}}$.

Remark 1.2 (Existence of Overlapping Samples)**.** *Consider two samples $\overline{x_i}$ and $\overline{x_j}$. Let $r_i = \max\{\gamma|\mathcal{V}_i|, \gamma|\mathcal{E}_i|\}$ and $r_j = \max\{\gamma|\mathcal{V}_j|, \gamma|\mathcal{E}_j|\}$. If $GED(\overline{x_i}, \overline{x_j}) < r_i + r_j$, then $\exists x' \in B(\overline{x_i}) \wedge x' \in B(\overline{x_j})$, i.e. at least one augmented sample belongs to both the induced augmentation sets.*

Definition 3 (Invalid Augmented Samples)**.** *We consider an augmented sample, x to be an invalid sample, if $x \in \mathcal{B}(\overline{x_i}) \wedge x \in \mathcal{B}(\overline{x_j})$ (an overlapping sampling), and $\mathbf{y}_i \neq \mathbf{y}_j$.*

Claim 1.1. *Given $\mathbf{D}, \Gamma, \mathbf{y}$, we can lower-bound the number of overlapping samples in the empirical data distribution as $\frac{1}{2}\sum_{i,j\in[1,\dots,|\mathcal{X}|]} \mathbb{1}(\mathbf{D}_{ij} - \Gamma_i - \Gamma_j \leq 0)$ where $\mathbb{1}$ is the indicator function. Furthermore, if we consider oracle label information, we can lower bound the number of invalid samples as $\frac{1}{2}\sum_{i,j\in[1,\dots,|\mathcal{X}|]} \mathbb{1}\big((\mathbf{D}_{ij} - \Gamma_i - \Gamma_j)|\mathbf{y}_i - \mathbf{y}_j| < 0\big)$.*

Proof. $\Gamma_i + \Gamma_j$ is the total number of edit operations that can be applied to either samples x_i or x_j. If the graph edit distance between samples i and j is smaller than this, then it is possible to reach the same augmented sample somewhere on the edit path that turns x_i into x_j regardless of which endpoint we start from. This augmented sample constitutes an overlapping sample, or an invalid sample if the class labels of x_i and x_j differ. Note that there may be multiple such augmented samples that can be created from either x_i or x_j; our indicator function counts one per pair of samples, and thus helps constitute a lower bound. □

1.1.2 Discussion on Invalid Samples

An invalid sample does not have a clear label because we do not know which natural label should be assigned to it. This can incur instability in discriminative methods if the invalid sample's loss is minimized with different labels over the course of training. It is also problematic for methods enforcing consistency because such methods will use the invalid sample to enforce consistency with respect to two different classes. We note that invalid samples will occur for any method that uses GGA and most methods will incur some irreducible error from training on an ambiguous sample.

Here, we discuss how inter-class and intra-class GED relate to number of invalid and overlapping samples. Let $\mathcal{I}$ be the set of all invalid samples, $\mathcal{O}$ the set of overlapping samples, and $\tilde{\mathcal{O}} := \mathcal{O} \setminus \mathcal{I}$ be the set of intra-class (valid) overlapping samples. Let C' be the lower bound on the number of invalid samples we computed in Claim 1.1. C' is controlled by two parameters, $\mathbf{D}$ and Γ. We see that whether samples are, on average, invalid or merely overlapping is dependent on the average distance between samples of different classes when Γ is held constant. Clearly, when training, we desire that $\frac{|\mathcal{I}|}{|\tilde{\mathcal{O}}|} \to 0$, as this ensures the model mostly sees valid samples. We note that this ratio is proportional to inter-class and intra-class distances as follows:

Recall that if $\mathcal{A}(\overline{x}_i) \in \mathcal{I}$, $(\mathbf{D}_{ij} - \Gamma_i - \Gamma_j)|\mathbf{y}_i - \mathbf{y}_j| < 0$ or equivalently, $\mathbf{D}_{ij} < \Gamma_i + \Gamma_j$, for $\mathbf{y}_i \neq \mathbf{y}_j$. Now, if $\mathcal{A}(\overline{x}_i) \in \tilde{\mathcal{O}}, \mathbf{D}_{ij} < \Gamma_i + \Gamma_j$, for $\mathbf{y}_i = \mathbf{y}_j$. Then, $|\mathcal{I}| \sim \mathbb{1}\left(\mathbf{D}_{ij} < \Gamma_i + \Gamma_j\right)$, for $\mathbf{y}_i \neq \mathbf{y}_j$ and $|\tilde{\mathcal{O}}| \sim \mathbb{1}\left(\mathbf{D}_{ij} < \Gamma_i + \Gamma_j\right)$, for $\mathbf{y}_i = \mathbf{y}_j$.

Now, $\frac{|\mathcal{I}|}{|\tilde{\mathcal{O}}|} \sim= \frac{\mathbb{1}(\mathbf{D}_{ij} < \Gamma_i + \Gamma_j), \text{ for } \mathbf{y}_i \neq \mathbf{y}_j}{\mathbb{1}(\mathbf{D}_{ij} < \Gamma_i + \Gamma_j), \text{ for } \mathbf{y}_i = \mathbf{y}_j} \to 0$, when inter-class distance is large for many samples, (i.e. the numerator is minimized), and when the intra-class distance is small for many samples (the denominator is maximized). This suggests it is desirable to have a lower average intra-class distance and a higher average inter-class distance.

While GED between samples cannot be controlled, the augmentation strength, Γ, can be controlled. It is desirable to minimize the number of invalid samples, while simultaneously maximizing the number of valid (including overlapping) augmented samples as follows:

$$\min_{\Gamma}(C) \text{ s.t. } \max_{\Gamma}\left(\sum_{\overline{x}\in\mathcal{X}} |\mathcal{B}(\overline{x})|\right)$$

While the above optimization is intractable and assumes label information, it alludes to two properties critical to the success of contrastive learning: connectedness of samples and recoverability (HaoChen et al., 2021). The number of invalid samples is indicative of the recoverability of different classes, while the above optimization indicates that we must also consider how well connected the augmentation sets are. We formalize this discussion in the next section.

1.1.3 GGA AND THE POPULATION AUGMENTATION GRAPH

The preceding section discusses the relationship between GGA, GED and error introduced by invalid samples. However, this analysis is method-agnostic and does not offer theoretical insights into graph contrastive learning.

In computer vision, recent attempts to analyze theoretically the performance of contrastive learning often assumes that sample views are independent, a condition clearly violated by data augmentation (Arora et al., 2019; Tosh et al., 2021). To avoid this assumption, HaoChen et al. (2021) recently introduced the notion of a *population augmentation graph* (PAG), which represents augmented samples as nodes and weighted edges as the likelihood of generating a given pair of augmented samples from the same clean sample. Because samples from the same class are more likely to produce the same augmented sample than two random classes, connected subgraphs or communities in the PAG naturally correspond to underlying classes. HaoChen et al. (2021) designed and theoretically analyzed a CL objective that performed spectral decomposition on the PAG to recover these subgraphs (classes). Using their proposed objective and the PAG, they were able to provide the first accuracy guarantees for CL.

We begin by defining the PAG and the assumptions critical to HaoChen et al. (2021)'s analysis. Then, we extend our analysis from the preceding section to discuss how well these assumptions are supported for GCL.

Definition 4 (Population Augmentation Graph (HaoChen et al., 2021))**.** *Given a natural dataset $\overline{\mathcal{X}}$, let $\mathcal{A}(\cdot|\overline{x})$ be the distribution of augmentations given a natural sample $\overline{x}$, or, intuitively, as the probability of generating a particular augmented sample from the large but finite set of all possible augmented versions of $\overline{x}$. Then, $\mathcal{X} := \cup_{\overline{x}\in\overline{\mathcal{X}}}\mathcal{A}(\cdot|\overline{x})$.*

Let $\mathcal{G}^p$ be the population augmentation graph, where all N samples in $\mathcal{X}$ form the nodes and $W \in \mathbb{R}^{N\times N}$ is the corresponding adjacency matrix. The edge weight between two nodes x and x' is defined as

$$w_{x,x'} := \mathbb{E}_{\overline{x}\in\mathcal{P}_{\overline{\mathcal{X}}}}[\mathcal{A}(x|\overline{x})\mathcal{A}(x'|\overline{x})].$$

Intuitively, if $w_{x,x'}$ is larger, it is relatively easier to generate the augmented pair from the same natural sample.

Now, since graphs are discrete, the augmentation severity is restricted and only one edit can be applied at a time, we can completely define the population augmentation graph. Specifically, by using Remark 1.1, the entire set of allowable augmentations can be determined. Moreover, recall that augmentations are performed randomly. Therefore, any $x \in \mathcal{B}(\overline{x}_i)$ is equally likely, so $\mathcal{A}(x|\overline{x}_i) = \frac{1}{|\mathcal{B}(x)_i|}$. However, if $x' \not\ni \mathcal{B}(\overline{x}_i)$, $\mathcal{A}(x|\overline{x}_i) = 0$ because it is not considered an allowable augmentation for $\overline{x}_i$. Note $w_{x,x'} > \frac{1}{|\mathcal{X}|}\frac{1}{|\mathcal{B}(\overline{x}_i)|^2}$ when x, x' are both *overlapping samples*, i.e. $x \in \mathcal{B}(\overline{x}_j), x' \in \mathcal{B}(\overline{x}_j)$ for $i \neq j$. We refer to an edge whose endpoints are both overlapping samples as an *overlapping edge*. Similarly, a node in the PAG that is an overlapping sample is referred to as an *overlapping node*. As such, we have defined all possible nodes in $\mathcal{G}^p$ as well as how the edges are defined.

1.1.4 EXPLORING PAG STRUCTURE

Claim 1.2. *(Node Degree) Let x be an overlapping node in the PAG. Additionally, suppose there is an alternative PAG, where $\tilde{x}$ is no longer an overlapping node but otherwise the PAG is the same. Then, x will have a larger degree than $\tilde{x}$. This is true even if $\tilde{x}$ is not in an overlapping edge.*

Proof. Because x is an overlapping node, $x \in \mathcal{B}(\overline{x}_i) \wedge x \in \mathcal{B}(\overline{x}_j)$ for some $i \neq j$. Then, $w_x = \sum_{x'} w_{xx'} = \sum_{x'\in\mathcal{B}(\overline{x}_i)} w_{xx'} + \sum_{x'\in\mathcal{B}(\overline{x}_j)} w_{xx'}$. Now, in the alternative PAG, $\tilde{x}$ is not an overlapping node, so $\tilde{x} \in \mathcal{B}(\overline{x}_i) \wedge \tilde{x} \notin \mathcal{B}(\overline{x}_j), \forall j \neq i$. Then $w_{\tilde{x}} = \sum_{x'} w_{\tilde{x}x'} = \sum_{x'\in\mathcal{B}(\overline{x}_i)} w_{xx'}$. Clearly, $w_x > w_{\tilde{x}}$. This that if a sample is an overlapping node, it will have a higher degree than if the same sample were not an overlapping node. □

Claim 1.3 (GED Influences PAG structure)**.** *If data points x, x' share an edge in the PAG, then* $\max\left(GED(x, \overline{x}_i), GED(x', \overline{x}_i)\right) < \max\{\gamma|\mathcal{V}_i|, \gamma|\mathcal{E}_i|\}$.

Proof. $w_{xx'} > 0$ if and only if $x \in \mathcal{B}(\overline{x}_i) \wedge x' \in \mathcal{B}(\overline{x}_i)$. Recall in Def. 1, that $x \in \mathcal{B}(\overline{x}_i)$ if and only if $GED(x, \overline{x}_i) < \max\{\gamma|\mathcal{V}_i|, \gamma|\mathcal{E}_i|\}$, and similarly for x'. □

Moreover, edge weights and node degrees are also influenced by the GED between samples. Overlapping edges can increase the weight between nodes. However, as discussed above, this requires that both ends of the edge are overlapping nodes. In Definition 2 and Remark 3, we show how GED can be used to determine the existence of such nodes. This further demonstrates the structure of the PAG is directly influenced by the GED between samples in $\mathcal{X}$.

We emphasize that our analysis suggests that practitioners may be using generic graph augmentations without realizing that they are implicitly assuming that GED is a useful metric for their problem.

Having elucidated the structure of the PAG and its relationships to GED, we discuss how its structure relates to the assumptions made by HaoChen et al. (2021) when analyzing the PAG. Namely, they require that the PAG "cannot be partitioned into too many disconnected sub-graphs", and that "labels are recoverable from augmentations." Indeed, their resulting bound on the error of spectral contrastive learning on the PAG depends upon the sparsest m-partition and classifier error.

The following assumption is from HaoChen et al. (2021):

Assumption 1. *(Labels are recoverable from augmentations). Let $\bar{x} \sim \mathcal{P}_{\overline{\mathcal{X}}}$ and $\mathbf{y}_{\bar{x}}$ be its label. Let the augmentation $x \sim \mathcal{A}(\cdot \mid \bar{x})$. We assume that there exists a classifier g that can predict $\mathbf{y}_{\bar{x}}$ given x with error at most α. That is, $g(x) = \mathbf{y}_{\bar{x}}$ with probability at least $1 - \alpha$.*

Claim 1.4. *(Recoverability is lower-bounded by the number of invalid samples). α (Assumption 1) can be lower-bounded when when $\mathcal{B}(\overline{x})$ contains an invalid sample: $\alpha \geq \frac{1}{\mathcal{B}(\overline{x})} - \frac{1}{|\mathcal{B}(x)|\tilde{Y}|}$, where where $\tilde{Y}$ is the set of labels represented among the natural samples that may have generated x.*

Proof. For this claim, we first discuss the best error that can be expected when classifying an invalid sample, and then we discuss the likelihood of encountering such a sample given some $\overline{x}$. Let x be an invalid sample that can be generated from natural samples: $\tilde{X} = \{(\overline{x}_1, \mathbf{y}_{\overline{x}_1}), \ldots (\overline{x}_k, \mathbf{y}_{\overline{x}_k})\}$. Clearly, x's label is not well defined as it could be assigned any label $\tilde{y} \in \tilde{Y}$. However, the classifier g, is assumed to predict $g(x) = \mathbf{y}_{\overline{x}_i}, \forall \overline{x}_i \in \tilde{X}$ with error at most α. Then, the minimum error for such a classifier is $1 - \frac{1}{|\tilde{Y}|}$, since the classifier does not know which natural sample generated x. For the remainder of the proof, we assume that g can correctly classify all samples expect invalid samples to derive a lower bound.

Having established the minimum error of a classifier on an invalid sample, we determine how likely g is to encounter such a sample given $\overline{x}$. Note that by assuming that g can correctly identify all other augmented samples, the classifier error is only incurred when x is an invalid sample. Through Remark 3, we first determine if an invalid sample is possible given a particular $\overline{x}$. If an invalid sample is possible, recall that every sample in the augmentation set, $\mathcal{B}(\overline{x})$, is equally likely by definition of generic graph augmentations. Therefore, the likelihood of generating x given $\overline{x}$ is $\sim \frac{1}{|\mathcal{B}(\overline{x})|}$, where the size of the augmentation set can be determined using Remark 1.1. We note that we could not provide an exact likelihood here because we assume (i) isomorphic graphs are counted separately and (ii) there is only one invalid sample in $\mathcal{B}(x)$, when in practice there may be multiple invalid samples. Nonetheless, we are able to derive a lower-bound on the error of the classifier, g, given a particular $\overline{x}$, by considering the likelihood of encountering an invalid sample and the error such a sample incurs: $\frac{1}{\mathcal{B}(\overline{x})} - \frac{1}{|\mathcal{B}(x)|\tilde{Y}|} \leq \alpha$. While the above analysis focus on a particular $\overline{x}$, we can extend the analysis to consider all samples, if we establish the likelihood of selecting a natural sample that can produce an invalid sample. (See subsubsection 1.1.2 for related discussion.) □

Lastly, we hypothesize that GED can be related to the Dirichlet conductance of the PAG, where Dirichlet conductance measures how many edges cross between a subset, S, and its complement relative to the total number of edges in the subset. We discuss our intuition in the following simple example, but leave a rigorous mathematical discussion to future work. Let $\overline{\mathcal{X}}$ be a dataset such that $\min_{i \neq j} GED(x_i, x_j) > \max(\Gamma)$, i.e the minimum distance between any two samples in the dataset is greater than the maximum allowable edits. Then, clearly the PAG contains $|\overline{\mathcal{X}}|$ fully connected

subgraphs (cliques) that correspond to $\mathcal{B}(\overline{x})$, where $w_{xx'} = \frac{1}{|\mathcal{B}(\overline{x})|}$ for $x, x' \in \mathcal{B}(\overline{x})$. Given the structure of the graph, the conductance is minimized when $S = \mathcal{B}(\overline{x})$, as all edges within the subset are already contained. There are no edges to the complement because there are no overlapping samples by construction. We suspect that this observation can be extended to understand the behavior of the sparsest m-partition of the PAG, which HaoChen et al. (2021) use in their error bounds, but we leave that analysis to future work.

1.2 Dataset Generation and Experimental Details

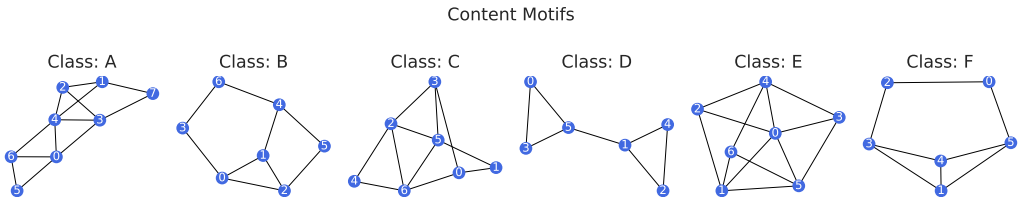

Figure 1: **Motifs used to determine class labels.**

We use the motifs shown in Fig. 1.2 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 (Xu et al. (2019)). 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 [1] 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 $\rho \in \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$ to generate the datasets used in Sec **??**.

Experimental Set-up: We follow You et al. (2020a) for TUDataset experiments. We use a 5-layer GIN model with sum pooling for all synthetic experiments. Models are pretrained for 100 epochs and then fine-tuned for 200 epochs with 1 learning rate drop when the loss plateaus. The hidden layer dimension is 32. We concatenate hidden representations for a representation dimension of 160. All models are trained with Adam, lr = 0.001. For the sample complexity experiment, we allow for end to end training. For all other experiments, we freeze the backbone and only train the linear prediction head.

1.3 Inductive Bias, Additional Results

To further demonstrate the effectiveness of untrained models on popular benchmarks, we include results from different GNN architectures: GraphSage (Hamilton et al. (2017)), PNA (Corso et al. (2020)), GCN (Kipf & Welling (2017)) and GAT (Velickovic et al. (2018)). We note that while it has been informally discussed that untrained GNNs have a strong inductive bias, our intention is to formalize why these untrained models *must* be included when evaluating unsupervised graph representation learning. Moreover, in Tab. 1.4, we include a variant of untrained models, where we initialize BatchNorm statistics without computing any gradients by iterating over the dataset once. For several datasets and baselines, we see that this Warmup step makes the untrained baseline even stronger. We believe that future work should also consider this simple baseline when evaluating the performance of their models.

1.4 Invariance, Additional Results

We extend our representation invariance results on standard benchmarks to different architectures below in Tab. 1.4. As in our main results, we use random subgraph sampling and node dropping

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

Table 2: Inductive Bias.

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	–

as our augmentations, following You et al. (2020a), when computing invariance. We find that similar trends hold: while training with GCL does improve performance and invariance somewhat, untrained models perform comparably without the same levels of invariance.

1.5 Dataset Statistics

1.6 Related Work

Graph Data Augmentation: Augmentations for graphs are difficult to define due to their discrete, non-euclidean nature. Furthermore, unlike images or natural language where there is an intuitive understanding of what changes will preserve task-relevant information, no such understanding exists 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. Zhao et al. (2020) note that a node classification task can be perfectly solved if edges only exist between same class samples. They 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. Kong et al. (2020) 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. To avoid incurring the large expense of adversarial training, they leverage Shafahi et al. (2019) and compute the gradient of the model weights while computing the gradients of the adversarial perturbation. This approach is not directly applicable to contrastive learning, where label information cannot be used to generate the adversarial perturbation.

Graph Self-Supervised Learning: Several paradigms for self-supervised learning in graphs have been recently explored, including the use of pre-text tasks, multi-tasks, and unsupervised learning. See Liu et al. (2021) for an up-to-date survey. Graph pre-text tasks are often reminiscent of image in-painting tasks Yu et al. (2018), and seek to complete masked graphs and/or node features (You et al. (2020b); Hu et al. (2020)). Other successful approaches include predicting graph level or property level properties during pre-training or part of regular training to prevent over-fitting (Hu et al. (2020)). These tasks often must be carefully selected to avoid negative transfer between tasks. Many

Table 3: Invariance Table.

	RandGAT	(Acc)	WarmupGAT	(Acc)	GAT (GraphCL)	(Acc)
MUTAG	0.993	0.843	0.364	0.793	0.608	0.807
PROTEINS	0.987	0.737	0.819	0.738	0.554	0.744
NCI1	0.993	0.761	0.543	0.771	0.669	0.781
DD	0.970	0.779	0.381	0.778	0.361	0.793
REDDIT-B	1.000	0.517	0.850	0.724	0.982	0.747
IMDB-B	1.000	0.512	0.979	0.670	0.994	0.512
	RandGIN	(Acc)	WarmupGIN	(Acc)	GIN (GraphCL)	(Acc)
MUTAG	0.921	0.867	0.208	0.866	0.852	0.868
PROTEINS	0.910	0.745	0.495	0.750	0.547	0.744
NCI1	0.921	0.707	0.281	0.769	0.768	0.778
DD	0.907	0.732	0.071	0.760	0.638	0.786
REDDIT-B	0.906	0.723	0.242	0.768	0.286	0.895
IMDB-B	0.914	0.672	0.791	0.700	0.468	0.711
	RandGCN	(Acc)	WarmupGCN		GCN (GraphCL)	(Acc)
MUTAG	0.996	0.847	0.491	0.807	0.561	0.821
PROTEINS	0.980	0.739	0.886	0.750	0.765	0.749
NCI1	0.991	0.756	0.480	0.767	0.664	0.780
DD	0.968	0.779	0.440	0.772	0.367	0.789
REDDIT-B	0.999	0.519	0.129	0.833	0.678	0.919
IMDB-B	0.914	0.540	0.539	0.833	0.994	0.709
	RandSAGE	(Acc)	WarmupSAGE	(Acc)	SAGE (GraphCL)	(Acc)
MUTAG	0.910	0.846	0.273	0.801	0.303	0.823
PROTEINS	0.907	0.732	0.582	0.747	0.507	0.749
NCI1	0.912	0.737	0.412	0.771	0.579	0.779
DD	0.590	0.771	0.590	0.781	0.727	0.801
REDDIT-B	0.833	0.849	0.225	0.740	–	–
IMDB-B	0.223	0.663	0.223	0.497	–	–

Table 4: *Dataset Description*

Name	**Graphs**	**Classes**	**Avg. Nodes**	**Avg. Edges**	**Domain**
IMDB-BINARY (Yanardag & Vishwanathan, 2015)	1000	2	19.77	96.53	Social
REDDIT-BINARY (Yanardag & Vishwanathan, 2015)	2000	2	429.63	497.75	Social
MUTAG (Kriege & Mutzel, 2012)	188	2	17.93	19.79	Molecule
PROTEINS (Borgwardt et al., 2005)	1113	2	39.06	72.82	Bioinf.
DD (Shervashidze et al., 2011)	1178	2	284.32	715.66	Bioinf.
NCI1 (Wale & Karypis, 2006)	4110	2	29.87	32.30	Molecule

unsupervised approaches have also been proposed. Sun et al. (2020); Velickovic et al. (2019) draw inspiration from Hjelm et al. (2019) and maximize the mutual information between global and local representations. MVGRL (Hassani & Ahmadi (2020)) contrasts different views at multiple granularities similar to van den Oord et al. (2018). You et al. (2020a); Qiu et al. (2020); Zhu et al. (2020); Thakoor et al. (2021); Kefato & Girdzijauskas (2021) use augmentations to generate views for contrastive learning. See Table 1.6 for a summary of the augmentations used. We note that random corruption, sampling or diffusion based approaches often do not preserve task relevant information or introduce meaningful invariances.

REFERENCES

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

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.

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

Table 5: **Selected Graph Contrastive Learning Frameworks.** Brief description of augmentations used by selected frameworks is provided. Most frameworks use random corruptive, sampling, or diffusion-based approaches to generate augmentations.

Method	Augmentations
GraphCL (You et al. (2020a))	Node Dropping, Edge Adding/Dropping, Attr. Masking, Subgraph Extraction
GCC (Qiu et al. (2020))	RWR Subgraph Extraction of Ego Network
MVGRL (Hassani & Ahmadi (2020))	PPR Diffusion + Sampling
GCA (Zhu et al. (2020))	Edge Dropping, Attr. Masking (both weighted by centrality)
BGRL (Thakoor et al. (2021))	Edge Dropping, Attr. Masking
SelfGNN (Kefato & Girdzijauskas (2021))	Attr. Splitting, Attr. Standardization + Scaling, Local Degree Profile, Paste + Local Degree Profile

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

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

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

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.

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.

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

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

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.

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

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. URL https://arxiv.org/abs/2103.00111.

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.

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.

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

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.

Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Velickovic, and Michal Valko. Bootstrapped representation learning on graphs. *CoRR*, abs/2102.06514, 2021. URL https://arxiv.org/abs/2102.06514.

Christopher Tosh, Akshay Krishnamurthy, and Daniel Hsu. Contrastive learning, multi-view redundancy, and linear models. In *Algorithmic Learning Theory, 16-19 March 2021, Virtual Conference, Worldwide*, volume 132 of *Proceedings of Machine Learning Research*, pp. 1179–1206. PMLR, 2021. URL http://proceedings.mlr.press/v132/tosh21a.html.

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

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

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

Nikil Wale and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. In *(ICDM*, 2006. doi: 10.1109/ICDM.2006.39.

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

Pinar Yanardag and S. V. N. Vishwanathan. Deep graph kernels. In *SIGKDD*, 2015. doi: 10.1145/2783258.2783417.

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

Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. When does self-supervision help graph convolutional networks? *CoRR*, abs/2006.09136, 2020b. URL https://arxiv.org/abs/2006.09136.

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. URL http://arxiv.org/abs/1801.07892.

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

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