Augmentations in Graph Contrastive Learning: Current Methodological Flaws & Towards Better Practices

Puja Trivedi  
University of Michigan  
pujat@umich.edu

Ekdeep Singh Lubana  
University of Michigan  
eslubana@umich.edu

Yaoqing Yang  
University of California, Berkeley  
yqyang@berkeley.edu

Danai Koutra  
University of Michigan  
dkoutra@umich.edu

ABSTRACT

Unsupervised graph representation learning is critical to a wide range of applications where labels may be scarce or expensive to procure. Contrastive learning (CL) is an increasingly popular paradigm for such settings and the state-of-the-art in unsupervised visual representation learning. Recent work attributes the success of visual CL to use of task-relevant augmentations and large, diverse datasets. Interestingly, graph CL frameworks report strong performance despite using orders of magnitude smaller datasets and employing domain-agnostic graph augmentations (DAGAs). Motivated by this discrepancy, we probe the quality of representations learnt by popular graph CL frameworks using DAGAs. We find that DAGAs can destroy task-relevant information and ham the model’s ability to learn discriminative representations. On small benchmark datasets, we show the inductive bias of graph neural networks can significantly compensate for this weak discriminability. Based on our findings, we propose several sanity checks that enable practitioners to quickly assess the quality of their model’s learned representations. We further propose a broad strategy for designing task-aware augmentations that are amenable to graph CL and demonstrate its efficacy on two large-scale, complex graph applications. For example, in graph-based document classification, we show task-relevant augmentations improve accuracy up to 20%.

CCS CONCEPTS

• Computing methodologies → Instance-based-learning.

KEYWORDS

Graph Neural Networks, Contrastive Learning, Data Augmentation

1 INTRODUCTION

Graph neural networks (GNNs) have been successfully used to learn representations for various supervised or semi-supervised graph-based tasks, including graph-based similarity search for web documents [18], fake news detection through propagation pattern classification [4, 12, 20, 44], activity analysis in web and social networks (e.g., discussion threads on Reddit, code repository networks on Github) [55], and scientific graph classification [25, 77, 84]. However, in many practical scenarios, labels are scarce or difficult to obtain. For example, web pages are seldom assigned with labels which summarize their contents, labeling fake news can be time-consuming, and labeling drugs according to their toxicity requires expensive wet lab experiments or analysis [15, 28, 29, 96]. Contrastive learning (CL) is an increasingly popular unsupervised graph representation learning paradigm for such label scarce settings [15, 22, 62, 86, 87] and is currently the state-of-the-art in unsupervised visual representation learning [6–8, 24].

Broadly, CL frameworks learn representations by maximizing similarity between augmentations of a sample (positive views) while simultaneously minimizing similarity to other samples in the batch (negative views). Recent theoretical and empirical works attribute the impressive success of visual CL (VCL) to two key principles: (i) leveraging strong, task-relevant data augmentation [52, 68, 76, 79, 95] and (ii) training on large, diverse datasets [3, 8, 24, 47, 51]. By using appropriate data augmentations, VCL frameworks learn high quality representations that are invariant to properties irrelevant to downstream task performance; thereby preserving task-relevant properties and preventing the model from learning brittle shortcuts [8, 52, 54, 68]. Large, diverse datasets are necessary as VCL frameworks routinely use 1K–8K samples in a batch to ensure that enough negative views are available to train stably [6, 8, 10, 24]. Representations learnt using VCL and self-supervised learning in general have been found to be more robust [26], transferable [30] and semantically aligned [57] than their supervised counterparts.

Interestingly, graph CL (GCL) frameworks often deviate from these key principles and yet report seemingly strong task performance. Small, binary graph classification datasets [45] are routinely used to benchmark GCL frameworks. Moreover, due to the non-euclidean, discrete nature of graphs, it can be difficult to design task-relevant graph data augmentations [38, 93] or know what invariances are useful for the downstream task. Therefore, frameworks often rely upon domain-agnostic graph augmentations (DAGAs) [87]. However, DAGAs can destroy task relevant information.
and yield invalid/false positive samples (see Fig. 1). It is also unclear if DAGAs induce invariances that are useful or semantically meaningful with respect to the downstream task.

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

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

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

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

For reproducibility, our code and data are available at https://github.com/GemsLab/GCAugmentations-BestPractices.

## 2 PRELIMINARIES & RELATED WORK

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

### 2.1 Contrastive Learning (CL)

**Frameworks & Losses.** Several CL frameworks [8, 21, 24] have been proposed to enforce similarity between positive samples and dissimilarity between negative samples, where positive samples are generated through data augmentation. Normalized temperature-scaled cross entropy (NT-XENT) is a popular objective used by several state-of-the-art CL frameworks [8, 61, 63, 71, 80, 86, 87] and is defined as follows. Let $X$ be a data domain, $D = \{x_1, \ldots, x_n \in X\}$ be a dataset, $T : X \rightarrow \tilde{X}$ be a stochastic data transformation and $f : (X, \tilde{X}) \rightarrow \mathbb{R}^d$ be an encoder. Further, assume we are given a batch of size $N$, similarity function $\text{sim} : (\mathbb{R}^d, \mathbb{R}^d) \rightarrow [0, 1]$, temperature parameter $\tau$, and encoded positive pair $\{z_i, z_j\}$. Then, NT-XENT can be defined as:

$$t_{i,j} = -\log \frac{\exp (\text{sim} (z_i, z_j) / \tau)}{\sum_{k=1}^{2N} \mathbb{I}_{\{k \neq i\}} \exp (\text{sim} (z_i, z_k) / \tau)}.$$  \hspace{1cm} (1)

Here, the numerator encourages the positive pair to be similar, while the denominator encourages negative pairs ($k \neq i$) to be dissimilar. Alternative CL objectives may enforce such (dis)similarity differently (e.g., through margin maximization [70] or cosine similarity [21]), but the principles discussed below uniformly explain the success of contrastive learning frameworks [2].

**The role of augmentations.** Recent work [52, 68, 69] has demonstrated that data augmentation is critical for training CL frameworks. Theoretically, Tian et al [68] show that positive views should preserve task-relevant information, while simultaneously minimalizing task-irrelevant information [69]. Training on such views introduces invariances to irrelevant information, leading to more generalizable representations. Indeed, state-of-the-art VCL frameworks [6, 8, 9, 17, 24] rely upon strong, task relevant data augmentation to generate such views. For example, Purushwalkam et al. [52] show that augmentations used by SimCLR introduce “occlusion invariance”, which is useful in classification tasks where objects may be occluded. Overall, we highlight that augmentation strategies are not universal [86, 87] and must align with the task; e.g., semantic segmentation tasks would benefit more from augmentations that induce view-point invariances [52].

**The role of large, high-quality datasets.** Empirically, CL frameworks [8, 10, 24] often require many negative samples in each batch to avoid class collisions (i.e., false positives) [2]. Further, recent theoretical work has shown that optimizing Eq. (1) is equivalent to learning an estimator for the mutual information shared between positive views, where the quality of this estimate is upper-bounded by batch-size [51, 71]. These properties combine to necessitate the use of large, diverse datasets in contrastive learning.
2.2 Graph Contrastive Learning (GCL)

Frameworks. In this paper, we focus on three state-of-the-art unsupervised representation learning frameworks for graph classification that represent different methodological perspectives: GraphCL [87], InfoGraph [62] and MVGRL [22]. Similar to SimCLR, GraphCL uses NT-XENT to contrast representations of augmented samples using a shared encoder. Much like DeepInfoMax [27], InfoGraph maximizes the mutual information between local and global views, where corresponding views are obtained through subgraph sampling and graph-pooling. Meanwhile, MVGRL mirrors CMC [67] and uses dual encoders to contrast multiple views of a graph, where views are generated by first running a diffusion process (e.g. Personalized Page Rank [48], Heat Kernel [37]) over the graph and then sampling subgraphs.

Graph data augmentation. Existing GCL frameworks leverage three main strategies to generate views: feature or topological perturbation (GraphCL), sampling (InfoGraph), and/or diffusion processes (MVGRL). We focus on the domain-agnostic graph augmentations (DAGAs) introduced by GraphCL, shown in Fig. 1, as these are more popular in recent frameworks [65, 86, 87], composable [8, 17], fast, and do not require dual view encoders. An empirical study on the benefits of DAGAs in GCL [87] demonstrates that (i) composing augmentations and adjusting augmentation strength to create a more difficult instance discrimination task improves downstream performance and (ii) augmentation utility is dataset dependent. However, a critical assumption underlying DAGA is that by limiting augmentation strength such that only a fraction of the original graph is modified, task-relevant information is not significantly altered. In Sec. 3, we revisit this assumption to show that it does not hold for many datasets and discuss the implications of training with poorly augmented graphs. Clearly, it is expected that models trained with task-aware augmentations (TAAs) that induce useful invariances will learn better features than those trained with DAGAs. However, graphs are often used as abstracted representations of structured data, such as molecules [96] or point clouds [59], and it is often unclear how to represent task-relevant invariances after abstracting to the graph space. In Sec. 4, we discuss a broad strategy for identifying augmentations that induce task-relevant invariances in the abstracted, graph space and demonstrate the significant performance boosts achieved by using such augmentations.

Automated Graph Data Augmentation. Concurrent works [23, 32, 33, 40, 50, 63, 86, 89] have begun investigating automated graph data augmentation as a means of both avoiding costly trial and error when selecting augmentations and generating more informative, task relevant views. These methods often use bi-level optimization objectives and/or viewmakers [64] to jointly learn representations and augmentations (cf. Appendix D for more details). Our analysis (Sec. 3) remains pertinent for GCL with automated augmentations. Namely, the proposed sanity check are not augmentation specific, the identified evaluation flaws must still be considered, and untrained models should still be included as baselines. Also, our discussion on the benefits and properties of TAAs (Sec. 4) remains relevant as it is difficult to identify post-hoc if an automated augmentation strategy is inducing semantically meaningful invariances or exploiting shortcuts.

3 REVISITING AUGMENTATIONS & EVALUATION IN GCL

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

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

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

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

Empirical Setup. In our analysis, we focus on commonly used graph classification datasets (Table 1) [45]. Official implementations for GraphCL, InfoGraph, and MVGRL are used. We consider the encoder architecture used by [87] and report results with graph convolutional layers from GIN [82] (original implementation), PNA [11], SAGE [19], GAT [73], and GCN [35]. See Appendix A for details on the training setup.

| Name          | Graphs Classes | Avg. Nodes | Avg. Edges | Domain     |
|---------------|----------------|------------|------------|------------|
| IMDB-BINARY   | 1000           | 2          | 19.77      | 96.53      | Social     |
| REDDIT-BINARY | 2000           | 2          | 429.63     | 497.75     | Social     |
| GOSIPCOP      | 5464           | 2          | 55.48      | 54.51      | News       |
| DEEEZER       | 9629           | 2          | 23.49      | 65.25      | Social     |
| GITHUB GDZER  | 12725          | 2          | 113.79     | 234.64     | Social     |
| MUTAG         | 188            | 2          | 17.93      | 19.79      | Molecule   |
| PROTEINS      | 1113           | 2          | 39.06      | 72.82      | Bioinf.    |
| DD            | 1178           | 2          | 284.32     | 715.66     | Bioinf.    |
| NCI1          | 4110           | 2          | 29.87      | 32.30      | Molecule   |

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

Given the importance of data augmentation in representation learning, several works [52, 68, 76, 81, 95] have investigated its properties. Recently, Gontijo-Lopes et al. [42] identified an empirical trade-off when selecting amongst augmentations to improve model generalization. Intuitively, augmentations should generate samples that are close enough to the original data to share task-relevant semantics and different enough to prevent trivially similar samples. This trade-off can be quantified through two metrics, affinity and diversity. Affinity measures the distribution shift between the augmented and original sample distributions. Diversity measures how difficult it is to learn from augmented samples instead of only training samples [42]. While augmentations that best improve generalization optimize for both metrics [42], it is not clear that DAGAs also optimize for both. For example, molecular graph classification tasks are commonly used to evaluate GCL frameworks. However, as noted in Fig. 1, limited perturbations are needed to invalidate a molecule or significantly alter its function. Here, augmented data is sufficiently diverse, but it is not clear if creating invalid molecule...
Table 2: Augmentation Affinity. Affinity [42], measured by the difference between original and augmented accuracy of a supervised model, captures how much the data distribution has changed as a result of augmentation. We see that DAGAs lead to low affinity. This is expected for molecular datasets, where it is easy to create invalid molecules, and is also true for some social network datasets.

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

samples also leads to low affinity, indicating that task-relevant information have been destroyed. We conduct the following experiment to understand the affinity of DAGAs on benchmark datasets.

Experimental setup. We measure affinity as follows: (i) train a supervised PNA encoder on the original training data, (ii) generate an augmented dataset by using random node/subgraph dropping at 20% of the graph size, as suggested by [87] and (iii) evaluate on clean and augmented training data separately. The difference between clean and augmented accuracy quantifies the distribution shift induced by augmentations [42].

Hypothesis. We argue that while it is not expected that accuracy on augmented data will match that of clean data, augmented accuracy should be nontrivial if augmentations are indeed information-preserving [76, 79].

Results. In Table 2, we see a considerable difference between clean and augmented accuracy across datasets. This implies low affinity, i.e., a large shift between augmented and training distributions, and confirms that DAGAs can destroy task-relevant information. Consequently, training on such samples will harm downstream task performance, as shown by prior works on VCL [79] and elucidated below for GCL.

3.2 (O2) Domain-agnostic augmentations induce weak discriminability

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

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

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

Results. In Fig. 2, we plot the normalized cosine similarity between representations (sorted by class label), such that the upper left and lower right quadrants correspond to the similarity between same-class representations. Results on additional datasets can be found in Appendix A. We see that MVGRL (Fig. 2d) and InfoGraph (Fig. 2c) are less likely to encounter false positive pairs as they, respectively, use diffusion-based views and maximize mutual information over sampled subgraphs. GraphCL, which uses DAGAs, is more likely to encounter false positive samples that can harm discriminative power (Obs. (O1)). Correspondingly, MVGRL and InfoGraph both
We aim to better contextualize the performance of GCL frameworks by conducting a systematic analysis of the inductive bias of GNNs, produce representations that are already discriminative without inductive bias and identify flaws in current GCL evaluation practices.

In Fig. 2, InfoGraph and MVGRL have similar task accuracy, but linear evaluation or task accuracy, which may hide differences in representations. Indeed, models are often only assessed through check for practitioners to assess the quality of their model’s learned representations with higher intra-class similarity than inter-class similarity. In contrast, GraphCL has low intra-class similarity as can be seen in the upper-left quadrant (Fig. 2b). This implies that the model has not learned features that capture the semantic similarity between the samples belonging to this class. However, we note that while MVGRL has learned discriminative representations, it requires dual encoders and it is unclear what invariances are learnt by training with diffusion-based views. Finally, we find that even though the randomly initialized, untrained model (Fig. 2a) has higher absolute values for average intra- and inter-class similarities than trained methods, it achieves inter-class similarity relatively lower than intra-class similarity, as required for discriminative applications. We further elaborate on this point in the next section.

Proposed evaluation practice. Given that CL frameworks directly optimize the similarity between representations, we argue that plotting representational similarity can serve as a simple sanity check for practitioners to assess the quality of their model’s learned representations. Indeed, models are often only assessed through linear evaluation or task accuracy, which may hide differences in the discriminative power of representations. For example, as shown in Fig. 2, InfoGraph and MVGRL have similar task accuracy, but MVGRL has learnt more discriminative representations.

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

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

As noted in Obs. (O2), randomly-initialized, untrained GNNs can produce representations that are already discriminative without any training (Fig. 2a). While the strength of inductive bias of GNNs in (semi-) supervised settings has been noted before [35, 56, 63, 91], we aim to better contextualize the performance of GCL frameworks by conducting a systematic analysis of the inductive bias of GNNs, using several datasets and architectures. Understanding the performance of untrained models helps contextualize the cost of training.

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

Results. As shown in Table 3, randomly-initialized, untrained models perform competitively against trained models on several benchmark datasets. It is likely that some of the negative effects of training with DAGAs (Obs. (O1)–(O2)) were mitigated by this strong inductive bias. However, note that it becomes difficult to justify the additional cost of GCL on datasets where task performance and representation quality are not noticeably better than untrained models. Below, we discuss how to fairly evaluate GCL frameworks and how popular benchmark datasets are, in fact, inappropriate for GCL.

Proposed evaluation practices. Given that randomly-initialized, untrained models are a non-trivial baseline for GCL frameworks, we argue that they should be included when evaluating novel frameworks to contextualize the benefits of unsupervised training. While some recent works [63, 83] include untrained models in their evaluation, this practice remains far from standardized.

Furthermore, CL frameworks often define negative samples through the other samples in the batch. Given the limited size of popular benchmark datasets (Table 3), it can be difficult to ensure that each batch is large enough to train stably. Further, given that these benchmarks are often binary classification tasks, half the samples, in a balanced setting, are expected to share the positive pair’s label but be treated as negative samples. This implies that representations learned with GCL may not be discriminative because models have minimized similarity for semantically related examples. We thus argue that evaluating GCL frameworks on these datasets is flawed and this practice should be discontinued.

We highlight that Dwivedi et al. [14] also find popular graph classification datasets are problematic in general, but for the specific case of GraphCL, this point is of some urgency as such small-scale datasets are part of standard GCL evaluation [63, 86]. However, we note that self-supervised frameworks that do not rely on negative samples, such as BYOL[17] and SimSiam[9], can be used as

| Dataset | (Samples) | Random Init (3 layers) | Random Init (4 layers) | Random Init (5 layers) | GraphCL [87] | InfoGraph [62] |
|--------|-----------|------------------------|------------------------|------------------------|-------------|-------------|
| IMDB-BINARY (1000) | 67.22 ± 7.77 | 61.26 ± 7.01 | 60.43 ± 5.92 | 71.14 ± 0.44 | 73.03 ± 0.87 |
| REDDIT-BINARY (2000) | 72.34 ± 6.64 | 64.57 ± 8.03 | 67.32 ± 7.41 | 89.53 ± 0.84 | 82.50 ± 1.42 |
| DEEZER (9629) | 56.59 ± 0.01 | 54.99 ± 1.74 | 54.87 ± 2.60 | 56.19 ± 0.015 | 55.89 ± 0.88 |
| GITHUB SGZR (12725) | 64.51 ± 0.05 | 64.93 ± 0.04 | 64.93 ± 0.82 | 65.81 ± 0.413 | Out of Time |
| MUTAG (188) | 85.76 ± 7.38 | 86.36 ± 6.51 | 86.73 ± 10.33 | 86.80 ± 1.34 | 89.01 ± 1.13 |
| PROTEINS (1113) | 73.64 ± 5.46 | 74.46 ± 4.09 | 74.22 ± 2.85 | 74.39 ± 0.45 | 74.44 ± 0.31 |
| DD (1178) | 73.23 ± 8.25 | 72.15 ± 7.25 | 72.08 ± 4.18 | 78.62 ± 0.40 | 72.85 ± 1.78 |
| NCI1 (4110) | 70.65 ± 1.99 | 70.36 ± 3.11 | 70.49 ± 2.42 | 77.81 ± 0.41 | 76.20 ± 1.06 |
We first focus on a binary graph-based document classification task. As shown by prior work [18], graph-based representations are effective at capturing not only the content but also the structure of a document, leading to improved classification performance in this setting. Here, our goal is to demonstrate adhering to VCL principles by using TAAs is needed to improve task performance.

Dataset & Task. The task is to classify movie reviews and plot summaries according to their subjectivity. Following [46], we convert the Subjectivity document dataset [49] (10k samples) into co-occurrence graphs, where nodes represent words, edges indicate that two words have co-occurred within the same window (e.g., window size 2 and 4), and node features are word2vec [43] embeddings. An example of this conversion is shown in Fig. 3a. Note that we only use positive-view-based self-supervised learning frameworks (e.g., SimSiam, BYOL) because this is a binary classification task (see Sec 3.3). Accuracy is computed using a kNN classifier.

Setup of GNN models. We use a Message Passing Attention Network [46] as the encoder, and a 2-layer MLP as the predictor. The representation dimension is 64, and models are trained using Adam [34] with LR=5e-4. Additional training details are given in Appendix B. We report results with the original GCN layer used by [46], as well as with GraphSAGE [19] and GIN [82] layers replacing it.

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

Task-Aware Augmentations. Recently, Wei et al. [78] proposed several intuitive augmentations for use in natural language processing, namely: synonym replacement, random word insertion, random word swapping and random word deletion, where the augmentation strength is determined by the sentence length. (See Fig. 3b for an example.) By design, these augmentations introduce invariances that are useful to downstream tasks (e.g., invariance to the occasional dropped word), preserve task-relevant information, and break view symmetry in the natural language modality. Due to a co-occurrence based construction process, changes in the underlying document will manifest in the corresponding graph, so it is likely that augmentations remain effective for the abstracted space.

Results. As shown in Table 4, task-relevant, natural language augmentations perform considerably better (up to +20%) than domain agnostic graph augmentations for both window sizes. Notably, TAAs are necessary to significantly improve performance over an untrained baseline, indicating that adhering to key principles of VCL is indeed beneficial.

Potential Graph Space Augmentations. While natural language augmentations modify samples prior to the graph construction process, it is easy to see that they can be converted into graph augmentations, effectively infusing DAGAs with domain knowledge on how to perturb co-occurrence graphs. Specifically, synonym replacement is equivalent to replacing node features of the selected word (node) with the closest word2vec embedding. Random insertion can be approximated in the co-occurrence graph by (i) creating a new node with a randomly selected word2vec embedding and (ii) duplicating the connections of an existing node. Random deletion can be represented by (i) randomly removing a node and (ii) rewiring the

3.4 Summary of Proposed Evaluation Practices

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

- Given that DAGA can destroy task-relevant information and harm the model’s ability to learn discriminative representations, there is need for designing context-aware graph augmentations (Sec. 4).
- Randomly initialized, untrained GNNs have strong inductive bias and should be reported during evaluation.
- Small, binary graph datasets are inappropriate for evaluating GCL frameworks.
- GCL frameworks should be comprehensively evaluated using metrics beyond accuracy to assess representation quality.

4 BENEFITS & DESIGN OF TASK-AWARE AUGMENTATIONS

In this section, we exemplify the benefits of adhering to key VCL principles by defining a broad strategy for finding task-aware augmentations in scenarios where prior domain knowledge is available. We note the goal of this strategy is not to resolve problematic data augmentations in GCL. Instead, we use the proposed strategy to help elucidate the benefits of abiding by VCL principles in two careful case studies.

Augmentation strategy: For many graph-based representation learning tasks, structured data, such as documents [46], propagation patterns [44], molecules [28], maps [31], and point-clouds [59], are first abstracted as graphs via a deterministic process before task-specific learning can begin. In this practical setting, our idea is to leverage knowledge pertaining to the original, structured data to find augmentations that will, in the abstracted graph space, (i) preserve task-relevant information, (ii) break view symmetry, and (iii) introduce semantically meaningful invariance. In our first case study, which focuses on a graph-based document classification task, we achieve this goal by exploiting existing natural language augmentations [78] and directly perturbing the raw input before its graph is constructed. However, when given a sufficiently complex graph construction process, it can be unclear if augmentations in the original space will induce useful invariances or retain task-relevant information in the abstracted graph space. In our second case study, which focuses on image classification using super-pixel nearest-neighbor graphs, we encounter this setting and propose to avoid destruction of task-relevant information by deliberately introducing task-irrelevant information. We then use augmentations designed to induce invariance to such irrelevant information.

4.1 Case Study 1: Document Classification

We first focus on a binary graph-based document classification task. As shown by prior work [18], graph-based representations are effective at capturing not only the content but also the structure of a document, leading to improved classification performance in this
with a dark and satirical look at new yorks wholesale garment industry, where shady deals are made for a buck and ruthlessness is a prerequisite for career success

with a dark and satirical look at new yorks wholesale garment industry, where shady deals are made for a dollar and meekness is a requirement for career success

(a) Original

(b) Natural Language Space

(c) Graph Space

Figure 3: Augmentations for Document Classification: Documents are represented as co-occurrence graphs [46, 92], where words are treated as nodes with word2vec embeddings, and edges indicate co-occurrence in a sliding windows [43]. As shown in (b), we perform synonym replacement (purple) and random word insertion (green) to augment sentences without losing task-relevant information [78]. In (c), we show random node (word) deletion (red). Our results show that natural language space augmentations improve classification accuracy substantially over baseline augmentations.

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

| Augmentation | GCN Siam Acc. | BYOL Acc. | SAGA Siam Acc. | BYOL Acc. | GIN Siam Acc. | BYOL Acc. |
|--------------|---------------|-----------|----------------|-----------|---------------|-----------|
| S. vs. S (ws = 2) | 69.41 ± 7.28 | 62.98 ± 3.12 | 59.17 ± 8.36 | 67.17 ± 2.70 | 55.67 ± 4.61 | 65.02 ± 2.00 |
| S vs. N (ws = 2) | 57.84 ± 4.31 | 65.78 ± 8.22 | 56.74 ± 1.70 | 63.77 ± 2.90 | 58.2 ± 8.24 | 74.26 ± 3.80 |
| Context-Aware (ws = 2) | 83.65 ± 2.31 | 78.12 ± 2.73 | 81.28 ± 2.54 | 78.23 ± 4.53 | 80.37 ± 4.07 | 77.79 ± 0.09 |
| S. vs. S (ws = 4) | 61.76 ± 5.12 | 66.38 ± 2.29 | 54.68 ± 1.53 | 67.37 ± 1.11 | 54.71 ± 3.00 | 66.18 ± 2.34 |
| S vs. N (ws = 4) | 55.38 ± 1.99 | 68.311.88 | 59.23 ± 8.03 | 70.6 ± 4.85 | 53.31 ± 1.36 | 66.59 ± 1.57 |
| Context-Aware (ws = 4) | 81.12 ± 3.97 | 74.05 ± 5.46 | 80.67 ± 10.36 | 75.65 ± 5.54 | 75.30 ± 15.61 | 76.55 ± 7.43 |

modified graph to connect neighbors of the removed node. Random swap is equivalent to swapping the features of two nodes. We highlight that domain-agnostic subgraph and node dropping do not rewire the co-occurrence graph. Thus, it is unclear what invariance to these augmentations represents in the original data modality. In Appendix B, we show that graph-space and document-space synonym replacement perform comparably, but leave the evaluation of other converted graph space augmentations to future work.

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

4.2 Case Study 2: Super-pixel Classification

Our second case study is based on super-pixel MNIST classification, a standard benchmark for evaluating GNN performance [14, 36]. Here, we pursue an alternative strategy for task-aware augmentation where augmentations must induce invariance to deliberately irrelevant information (e.g., color for digit classification).

Dataset & Task. We follow the established protocols in [14, 36] to create super-pixel representations of MNIST, where each image is represented as a k-nearest neighbors graph between super-pixels (homogeneous patches of intensity). Nodes map to super-pixels, node features are super-pixel intensity and position, and edges are connections to k neighbors. An example is shown in Fig. 4.

Setup of GNN models. The following architecture is used for experiments. The encoder is 5-layer GIN architecture similar to [87] and [14]. The projector is a 2-layer MLP and there is no predictor. Models are trained for 80 epochs, using Adam [34] with LR of 1e-3, and the representation dimension is set to 110. The models are trained using SimSiam [9], BYOL [17], and SimCLR [8]. We give more training details in Appendix C. While composing augmentations is known to improve performance on vision tasks, we avoid it here in order to fairly compare to graph baselines, which only consider a single augmentation.
Table 5: Super-pixel Classification. KNN Accuracy after unsupervised training with Node Dropping (ND) or context aware graph augmentations (Colorize) is reported. Context aware augmentations improve performance. Accuracy of randomly initialized model is 37.79 ± 0.03.

| Aug.      | SimSiam Acc. | SimCLR Acc. | BYOL Acc. |
|-----------|--------------|-------------|-----------|
| ND (20%)  | 66.30 ± 0.33 | 68.56 ± 0.16 | 65.32 ± 0.95 |
| ND (30%)  | 61.30 ± 0.48 | 68.07 ± 0.37 | 61.87 ± 1.03 |
| Colorize  | 68.95 ± 1.20 | 73.67 ± 0.10 | 64.42 ± 2.385 |

Domain-Agnostic Graph Augmentations. Following [87], we apply random node dropping at 20% of the graph size to obtain both samples in the positive pair.

Task-Aware Augmentations. While geometric image augmentations [8], such as horizontal flipping and rotating, generally preserve task-relevant information and introduce semantically meaningful invariance, they cannot break view symmetry in GCL frameworks as GNNs are permutation invariant. Therefore, the representations of a pair of flipped images will be similar as their corresponding super-pixel graph representations are equivalent up to node reordering. On the other hand, augmentations such as cropping may result in qualitatively different super-pixel graphs. Here, it is unclear if the super-pixel graph obtained after augmentation preserves task-relevant information, even if cropping is information preserving with respect to the original image. Therefore, it is not trivial to identify successful augmentations in the abstracted domain that will also be successful in graph space.

Given the difficulty of identifying augmentations that perturb super-pixel graph topology but also preserve task-relevant information, we focus on image space augmentations that lead to modified node features in the super-pixel graph. Specifically, we select random colorization as the TAA as it (i) preserves task-relevant information as color is not relevant property when classifying digits, (ii) breaks view symmetry because the node features of augmented samples are different and (iii) introduces a harmless invariance to colorization. We briefly note that augmentations are generally selected to introduce invariances that are useful to the downstream task. For example, cropping results in occlusion invariance, which is useful for classification tasks where objects are often partially covered [52]. Here, we take a complementary approach where augmentations introduce harmless information (color) and the model learns to ignore it. This can be a useful strategy when it is difficult to clearly identify potentially useful invariances for a given task.

Results. In Table 5, we observe that training with an information-preserving, TAA (colorizing) improves accuracy for both SimSiam and SimCLR. While BYOL generally performs worse than SimSiam and SimCLR, colorizing is still within standard deviation of DAGAs. Composing augmentations with colorizing would likely further improve performance, but this investigation is left to future work. This confirms that learning invariance to irrelevant information, as determined by knowledge of the original data modality, is indeed a viable strategy for creating TAAs. Moreover, we note that randomly-initialized models have 37.79% accuracy, indicating that super-pixel data can serve as a sufficiently complex benchmark for future GCL evaluation [14] (see Appendix C for affinity and representational similarity analysis).

5 CONCLUSION

In this work, we discuss limitations in the evaluation and design of existing instance-discrimination GCL frameworks, and introduce new improved practices. In two case studies, we show the benefits of adhering to these practices, particularly the benefits using task-aware augmentations. First, through our analysis, we show that domain-agnostic graph augmentations do not preserve task-relevant information and lead to weakly discriminative representations. We then demonstrate that benchmark graph classification datasets are not appropriate for evaluating GCL frameworks by contextualizing recent theoretical work in VCL. Indeed, we show that the strong inductive bias of randomly initialized, untrained GNNs obfuscates GCL framework inefficiencies. While we acknowledge the community is moving toward larger and more extensive benchmarks [14], we emphasize that it is fundamentally incorrect to continue evaluating GCL on legacy graph classification benchmarks. Furthermore, on two case studies with practically complex tasks, we show how to use domain knowledge to perform information-preserving, task-aware augmentation and achieve significant improvements over training with domain-agnostic graph augmentations. In summary, GCL is an exciting new direction in unsupervised graph representation learning and our work can inform the evaluation of new methods as well as help practitioners design task-aware augmentations.

ACKNOWLEDGMENTS

We thank Jay Thiagarajan and Mark Heimann for helpful discussions on the project. This work is partially supported by the National Science Foundation under CAREER Grant No. IIS 1845491, Army Young Investigator Award No. W911NF180397, and Adobe, Amazon, Facebook, and Google faculty awards. Any opinions, findings, and conclusions or recommendations expressed here are those of the author(s) and do not reflect the views of funding parties.
A EXPERIMENTAL DETAILS OF SECTION 3

For Secs. 3.2, 3.3 experiments, we use a GIN-based encoder [82] similar to InfoGraph [62] and GraphCL [87] for all datasets but (DEEZER, GOSSIPCOP, GITHUB-SGZR). PNA is used for (DEEZER, GITHUB-SGZR) to stabilize Infograph’s loss and in Sec. 3.1. For GOSSIPCOP, the encoder is based off PyG’s implementation [16]: 1 GCN Layer, 1 Linear Layer, embedding dimension = 128, Optimizer = Adam [34], LR = 0.001, # of Epochs = 30, Batch-Size = 32. Models are trained on a Nvidia Tesla K80 GPU with Adam. A batch-norm layer is included between the output of the backbone and cross entropy layer. For augmentations, we follow [87] and stochastically apply node dropping at 20% of graph size and subgraph dropping at 20% of graph size.

Sec. 3.2 Experimental Setup: 3-layer GIN model with hidden dimension, learning rate, and epochs trained of (32, NA, NA) for RAND (Random Initialization), (512, 0.001, 20) for InfoGraph, and (32, 0.01, 20) for GraphCL. Adam and Nvidia Tesla K80 GPUs (12-GB GPU) were used to train all models. Results for MVGRL ([22]) are not included as we consistently witnessed Out-Of-Memory errors. Results are reported over 3 seeds. Additional Results: Fig. 5a includes additional results for PROTEINS, NCI1 and DD datasets.

Sec. 3.3 Experimental Setup: For all datasets, excluding DEEZER and GITHUB-SGZR, we report results from GraphCL and InfoGraph. We use the same GIN encoder as GraphCL when reporting the performance of randomly initialized models for these datasets. On GITHUB-SGZR, InfoGraph training time on exceeds eight hours using a NVIDIA TESLA P100. Additional Results: See Table 6. We find that the inductive bias of GNNs is strong across different architectures (GraphSAGE, PNA, GCN, and GAT).

Table 6: Inductive Bias: Additional results.

| Graph | NotAug | 3 Layer | 4 Layer | 5 Layer | GraphCL | InfoGraph |
|-------|--------|---------|---------|---------|---------|-----------|
| MUTAG | 0.85 ± 0.005 | 0.85 ± 0.006 | 0.85 ± 0.005 | 0.82 ± 0.004 | 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 ± 0.005 | 0.74 ± 0.011 | 0.78 ± 0.006 | 0.79 ± 0.014 |
| DD | 0.77 ± 0.006 | 0.78 ± 0.002 | 0.78 ± 0.005 | 0.80 ± 0.008 | 0.77 ± 0.011 |
| REDDIT-B | 0.85 ± 0.014 | 0.83 ± 0.016 | 0.83 ± 0.005 | – | 0.66 ± 0.157 |
| IMDb-B | 0.66 ± 0.012 | 0.81 ± 0.008 | 0.81 ± 0.008 | – | – |
| PNA | 0.88 ± 0.011 | 0.88 ± 0.009 | 0.89 ± 0.009 | 0.86 ± 0.023 | 0.90 ± 0.014 |
| MUTAG | 0.88 ± 0.011 | 0.88 ± 0.009 | 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 | 0.66 ± 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 |

Fig. 5: Representational Similarity: In addition to MUTAG (Figure 2), we provide results on PROTEINS, NCI1 and DD. Random inductive bias is most noticeable on MUTAG and PROTEINS. Note that the in-class similarity can be low for GraphCL and InfoGraph.

B DOCUMENT CLASSIFICATION

In Sec. 4.1, we demonstrate the benefits of using task-aware augmentations on a graph-based document classification task.

Experimental Setup: We use the model, code base and default settings of [46]. Models are trained using Adam: lr = 0.001, weight-decay = 1e-4 and cosine scheduler (T=8). We use the code (https://github.com/jasonweiz/eda-nlp) and augmentations by [78]. Synonym replacement, random deletion, random insertion and random swapping are applied at 5%, 10%, 5%, 5% of sentence length respectively. We generate an augmented version of each sentence for every training epoch. For domain agnostic augmentations, we apply random node dropping (10%) to generate one view. The other view is generated by applying random node or subgraph dropping (10%).

As noted in Sec. 4.1, natural language augmentations can be directly in graph space. We provide proof of concept using the synonym replacement augmentation. In Table 7, results are reported for a model trained with synonym replacement and graph space equivalent, node replacement at 5%. This model achieves comparable accuracy to the original task-aware augmentations. We suspect that synonym replacement is crucial for this task.
C SUPER-PIXEL CLASSIFICATION

In Sec. 4.2, we demonstrate the benefits of using task-aware augmentations via a case study on MNIST superpixel classification.

**Experimental Setup:** 50K images are used for training, 10K for validation, and 10K for testing. We follow the same procedure as [14] to convert images to superpixel graphs: SLIC ([1]) is used to extract superpixels from the image. Then, a kNN graph is constructed between the superpixels. Node features are RGB values and (x, y) coordinates of superpixels. Classification is performed using three hyper-parameters and architecture are used for all frameworks. Specifically, we use a 5-Layer GIN model closely following [14]. This model is converted from DGL ([https://www.dgl.ai]) to PyG ([16]). The following hyper-parameters are used: LR=5e-4, Hidden-Dim =110, Epochs=80, Batch-size = 128. The Adam ([34]) Optimizer is used for automatic data-augmentation, some of which we briefly describe in Table 11.

### Table 7: Document Classification

We use the same augmentations as in Table 4. Text-to-Graph augmentations perform synonym replacement as modifying features.

| Augmentation                        | SimCLR KNN Acc. | BYOL KNN Acc. |
|-------------------------------------|-----------------|--------------|
| S. vs S. (ws = 2)                   | 62.62 ± 3.21    | 66.25 ± 2.65 |
| S. vs N. (ws = 2)                   | 57.35 ± 2.47    | 62.83 ± 2.82 |
| Text-Space (ws = 2)                 | 83.69 ± 0.01    | 82.69 ± 1.98 |
| Text-to-Graph (ws = 2)              | 83.33 ± 1.29    | 78.16 ± 2.11 |
| S. vs S. (ws = 4)                   | 63.70 ± 0.71    | 65.73 ± 0.50 |
| S. vs N. (ws = 4)                   | 54.77 ± 1.42    | 65.99 ± 2.78 |
| Text-Space (ws = 4)                 | 83.29 ± 0.9     | 72.91 ± 4.97 |
| Text-to-Graph Space (ws = 4)        | 84.67 ± 1.37    | 77.96 ± 2.04 |

### Table 8: Comparison to [75]. Results only reported for SimCLR, as it performs better than SimSiam and BYOL in preceding experiments.

| Rand Init. | ND (20%) | ND (30%) | Colorize | DACL [75] |
|------------|----------|----------|----------|-----------|
|            | ±0.03    | ±0.16    | ±0.37    | ±0.10     | 59.94 ± 0.01 |

### Table 9: Super-pixel, Rep. Similarity. Avg. intraclass and interclass cosine similarity is reported. Coloring produces representations with the largest difference between intra- vs. inter-class similarity, indicating that representations are well-separated.

| Method | Aug. | Intra Sim | Inter Sim | Abs. Diff | Rel. Diff | Acc. |
|--------|------|-----------|-----------|-----------|-----------|------|
| SimCLR | ND (20%) | 87.71 | 78.62 | 8.44 | 0.0928 | 68.56 ± 0.16 |
| SimCLR | ND (30%) | 87.01 | 79.61 | 7.84 | 0.0911 | 68.07 ± 0.37 |
| SimCLR | Colorizing | 80.801 | 67.812 | 12.988 | 0.1607 | 73.67 ± 0.10 |

### Table 10: Super-pixel Affinity. Supervised, clean train accuracy is 90.01% and clean test accuracy is 88.69%.

| Aug. | Aug. Train Acc. | Aug. Test Acc. |
|------|-----------------|----------------|
| ND (20%) | 39.42 ± 0.011 | 40.29 ± 0.054 |
| ND (30%) | 29.19 ± 0.001 | 29.09 ± 0.036 |
| Colorizing | 47.86 ± 0.010 | 48.97 ± 0.035 |

### Table 11: Selected GCL Frameworks

| Method | Augmentations |
|--------|---------------|
| BGRIL [66] | Edge Dropping, Attr. Masking |
| GCA [94] | Edge Dropping, Attr. Masking (both weighted by centrality) |
| GCC [11] | BVR Subgraph Extraction of Egg Network |
| GraphCl [87] | Node Dropping, Edge Adding/Dropping, Attr. Masking, Subgraph Extraction |
| MVGRL [22] | PPR Diffusion + Sampling |
| SelfGNN [12] | Attr. Splitting, Attr. Standardization + Scaling, Local Degree Profile, Paste + Local Degree Profile |
| JOAO [36] | Min Max Optimization to adaptively and dynamically select from DAGA set |
| GraphAugmentor [33] | Learnable Feature Augmentors that can be applied per/post encoding |
| BYOV [89] | Uses graph generation (regularized by InfoMin + InfoBottleNeck) as viewmaker |
| AdvGCL [64] | Adversarial/MinMax Optimization over learnable augmentations |
| AF-GRL [46] | Finds node-level positive samples sharing "local structure and global semantics" |
| LGZAR [23] | Learns a policy over augmentations and their respective strengths without bi-level optimization |