An Empirical Study of Graph Contrastive Learning

Yanqiao Zhu[1,2] **Yichen Xu**[3] **Qiang Liu**[1,2] **Shu Wu**[1,2]
[1]Center for Research on Intelligent Perception and Computing
Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]School of Computer Science, Beijing University of Posts and Telecommunications
`yanqiao.zhu@cripac.ia.ac.cn`, `linyxus@bupt.edu.cn`
`{qiang.liu, shu.wu}@nlpr.ia.ac.cn`

Abstract

Graph Contrastive Learning (GCL) establishes a new paradigm for learning graph representations without human annotations. Although remarkable progress has been witnessed recently, the success behind GCL is still left somewhat mysterious. In this work, we first identify several critical design considerations within a general GCL paradigm, including augmentation functions, contrasting modes, contrastive objectives, and negative mining techniques. Then, to understand the interplay of different GCL components, we conduct extensive, controlled experiments over a set of benchmark tasks on datasets across various domains. Our empirical studies suggest a set of general receipts for effective GCL, e.g., simple topology augmentation that produces sparse graphs brings most performance improvements; contrasting modes should be aligned with the granularities of end tasks. In addition, to foster future research and ease the implementation of GCL algorithms, we develop an easy-to-use toolbox PyGCL, featuring modularized CL components, standardized evaluation, and experiment management. We envision this work to provide useful empirical evidence of existing GCL architectures and offer several insights for future research.

1 Introduction

The past few years have seen rapid advances in Graph Neural Networks (GNNs) [1, 2], which is a powerful framework for analyzing graph-structured data. As the most GNN models focus on (semi-)supervised learning, which requires access to abundant labels, recent trends in Self-Supervised Learning (SSL) have led to a proliferation of studies that learns graph representations without relying on human annotations. Among SSL methods, Contrastive Learning (CL), also known as instance discrimination, is a major area of interest and has already achieved comparable performance with its supervised counterparts in many representation learning tasks [3–12].

Recently, remarkable progress has been made to adapt CL techniques for the graph domain. A typical Graph Contrastive Learning (GCL) framework firstly constructs multiple graph views via stochastic augmentation of the input, and then learns representations by contrasting positive samples against negative ones. For each node being an anchor instance, positive samples are often chosen as its congruent representations in other views, while negatives are selected from other nodes within the given graph in node-centric datasets or other graphs in graph-oriented datasets. Although GCL has constituted a new paradigm of SSL in the graph domain and achieved promising results, recent studies [13–19] seem to resemble each other with very limited nuances from the methodological perspective. Moreover, most existing work only provides model-level evaluation. Still, the contributing factors leading to the success of GCL remain somewhat mysterious, which calls for a deeper understanding of different GCL components.

Preprint. Under review.

Towards this end, we try to shed light on the success behind these GCL algorithms through the lens of empirical evaluation of critical design considerations in existing work. We first propose a general contrastive paradigm and characterize previous work by limiting the design space of interest to four dimensions: (a) data augmentation functions, (b) contrasting modes, (c) contrastive objectives, and (d) negative mining strategies. Note that we include no model-specific design considerations such as the number of attention heads for graph attentive encoders. To the best that we are aware of, these four dimensions cover a wide range of options that are representative in a large body of literature.

Then, we systematically study the empirical performance of different design dimensions through controlled experiments over three benchmark tasks on a set of datasets across a variety of domains. Our empirical studies attempt to provide answers to the following questions:

- What is the most contributory component in an effective graph CL algorithm?
- How do different design considerations affect the model performance?
- Do these design considerations favor certain types of data or end tasks?

We note that although there has been several survey papers on self-supervised graph representation learning [20–22], to the best of our knowledge, none of existing work provides rigorous empirical evidence on the impact of each component in GCL.

We summarize several key findings of the empirical study, which we hope could benefit the graph SSL community for developing future algorithms. Our experiments suggest a set of general recipes for effective GCL algorithms:

- GCL algorithms benefit most from topology augmentations that produce sparse graph structures. In addition, bi-level augmentation on both topology and feature levels further improves the performance, if informative node attributes are available.
- The contrasting modes should be chosen according to the granularity of downstream tasks.
- The InfoNCE objective achieve stable, consistent performance improvements under all settings yet requires a large number of negative samples.
- Several recently proposed negative-sample-free objectives have great potential in reducing computational burden, while obtaining promising performance.
- Current negative mining strategies bring limited performance improvements to GCL.

In addition, to foster future research, we develop PyGCL, an easy-to-use PyTorch framework, featuring commonly used, modularized GCL components, standardized evaluation, and experiment management. We hope the use of PyGCL will greatly relief the burden of comparing existing baselines and developing new algorithms. The PyGCL is open-sourced at `https://github.com/GraphCL/PyGCL`.

2 A General Paradigm of GCL and its Design Dimensions

Problem formulation. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a given graph, where $\mathcal{V} = \{v_i\}_{i=1}^{N}$, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represent the node set and the edge set respectively. We further denote the feature matrix and the adjacency matrix as $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ and $\boldsymbol{A} \in \{0,1\}^{N \times N}$, where $\boldsymbol{x}_i \in \mathbb{R}^{F}$ is the feature of v_i, and $\boldsymbol{A}_{ij} = 1$ iff $(v_i, v_j) \in \mathcal{E}$. In the setting of unsupervised graph representation learning, there is no given class information of nodes in $\mathcal{G}$ during training. Our objective is to learn a GNN encoder f receiving the graph features and structure as input, that produces node embeddings in low dimensionality. We denote $\boldsymbol{H} = f(\boldsymbol{X}, \boldsymbol{A}) \in \mathbb{R}^{N \times F'}$ as the learned representations of nodes, where $\boldsymbol{h}_i$ is the embedding of node v_i. For graph-oriented tasks, we can further obtain a graph-level representation $\boldsymbol{s} = r(\boldsymbol{H}) \in \mathbb{R}^{F'}$ of $\mathcal{G}$ that aggregates node-level embeddings. Note that the readout function r might be a simple permutation-invariant function such as mean or sum pooling, or may be learnable and parameterized by a neural network. These representations can be used in downstream tasks, such as node/graph classification and community detection.

General paradigm of GCL. We decompose representative GCL algorithms from four dimensions: (a) data augmentation functions, (b) contrastive mode, (c) contrastive objective, and (d) negative mining strategies. These four components constitute the design space of interest for GCL in this work.

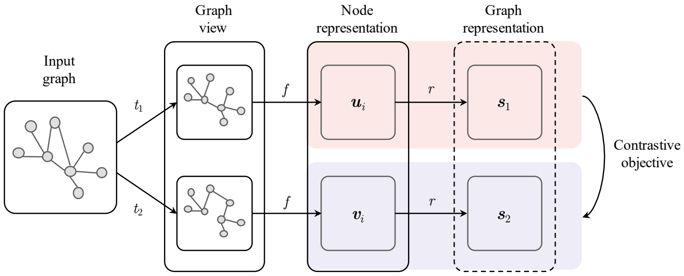


Figure 1: A general GCL model. At first, two graph views are generated via data augmentation functions. Then, the two graphs are fed into a shared graph neural network to learn representations, which are then optimized with a contrastive objective that pulls congruent representation pairs in the two views together while pushing others away. Additional negative sampling techniques may be employed to improve the model performance.

At each iteration of training, we first perform stochastic **augmentation** to generate multiple graph views from the input graph. Specifically, we sample two augmentation functions $t_1, t_2 \sim \mathcal{T}$ to generate graph views $\widetilde{\mathcal{G}}_1 = t_1(\mathcal{G})$ and $\widetilde{\mathcal{G}}_2 = t_2(\mathcal{G})$, where $\mathcal{T}$ is the set of all possible transformation functions to be discussed in the next section. We then obtain node representations for the two views using the shared GNN encoder f, denoted by $\boldsymbol{U} = f(\widetilde{\boldsymbol{X}}_1, \widetilde{\boldsymbol{A}}_1)$ and $\boldsymbol{V} = f(\widetilde{\boldsymbol{X}}_2, \widetilde{\boldsymbol{A}}_2)$ respectively. Optionally, we may obtain graph representations for each graph view: $\boldsymbol{s}_1 = r(\boldsymbol{U})$ and $\boldsymbol{s}_2 = r(\boldsymbol{V})$.

For every node embedding $\boldsymbol{v}_i$ being the anchor instance, the **contrasting mode** specifies a positive set $\mathcal{P}(\boldsymbol{v}_i) = \{\boldsymbol{p}_i\}_{i=1}^{P}$ and a negative set $\mathcal{Q}(\boldsymbol{v}_i) = \{\boldsymbol{q}_i\}_{i=1}^{Q}$. In a pure unsupervised learning setting, we only consider congruent samples in each graph view; in other words, embeddings in the two augmented graph views corresponding to the same node or graph constitute the positive set. Thus, we denote the only positive sample as $P(\boldsymbol{v}_i)$ afterwards for simplicity. It is noted that when label supervision is given, the positive set may be enlarged with samples belonging to the same class [23]. Moreover, we may employ **negative mining strategies** to improve the negative sample set by considering the relative similarity (i.e. the hardness) of negative samples. Finally, we use a **contrastive objective** $\mathcal{J}$ to score these specified positive and negative pairs and train the model in an unsupervised manner.

In the following we succinctly list implementations of these four design dimensions considered in this work. For details of each implementation, we refer readers of interest to Appendix E.

2.1 Design Dimensions

Data augmentation. The purpose of data augmentation is to generate *congruent, identity-preserving positive samples* of the given graph. Most graph CL work involves bi-level augmentation techniques: topology (structure) transformation and feature transformation. In this work we consider eight topology augmentation functions and two feature augmentations.

- **Topology augmentation:** (1) Edge Removing (ER), (2) Edge Adding (EA), (3) Edge Flipping (EF), (4) Node Dropping (ND), (5) Ego-net Sampling (ES), (6) Subgraph induced by Random Walks (RWS), (7) diffusion with Personalized PageRank (PPR), and (8) diffusion with Markov Diffusion Kernels (MDK).
- **Feature augmentation:** (1) Feature Masking (FM) and (2) Feature Dropout (FD).

Contrasting modes. For an anchor instance, contrasting modes determine the positive and negative sets at different granularities of the graph. In mainstream work, three contrasting modes are widely employed: (1) local-local CL and (2) global-global CL, which contrast embeddings at the same scale, and (3) global-local CL, which contrasts cross-scale embeddings.

Contrastive objectives. Contrastive objectives are used to train the encoder to maximize the agreement between positive samples and the discrepancy between negatives. We consider the following objective functions in this work: (1) InfoNCE, (2) Jason-Shannon Divergence (JSD), and (3) Triplet Margin loss (TM).

Table 1: Summary of representative graph CL models within the proposed paradigm.

Method	Primary task	Topology augmentation	Feature augmentation	Contrasting mode	Dual branches?	Contrastive objective
DGI [13]	Node classification	ES	—	Global-local	✗	JSD
GMI [15]	Node classification	ES	—	Global-local	✗	SP-JSD
InfoGraph [19]	Graph classification	ES	—	Global-local	✗	SP-JSD
MVGRL [14]	Node & graph classification	PPR	—	Global-local	✓	JSD
GCC [24]	Transfer learning	RWS + ES	—	Local-local	✗	InfoNCE
GraphCL [16]	Node & graph classification	RWS/ND/EA/ED	FD	Local-local	✓	InfoNCE
GRACE [17]	Node classification	ER	MF	Local-local	✓	InfoNCE
GCA [18]	Node classification	ER	MF	Local-local	✓	InfoNCE
BGRL [25]	Node classification	ER	MF	Local-local	✓	BL
GBT [26]	Node classification	ER	MF	Local-local	✓	BT

Negative mining strategies. Most existing work presumes embeddings of nodes or graphs other than the anchor instance to be dissimilar to the anchor and thus considers them as negatives. In this work, we consider the following four negative mining schemes: (1) Hard Negative Mixing (HNM), (2) Debiased Contrastive Learning (DCL), (3) Hardness-Biased Negative Mining (HBNM), and (4) Conditional Negative Mining (CNM).

2.2 Discussions on Representative GCL Methods

We give a brief summary of existing representative GCL methods as shown in Table 1 and discuss how these implementations fit into our proposed paradigm. It should be noted that negative mining strategies have received scant attention in current GCL literature and thus are omitted in the table.

Dual branches vs. single branch. We notice that most work leverages a dual-branch architecture following SimCLR [8] that augments the original graph twice to form two views and designates positive samples across two views. For some global-local CL methods like DGI [13] and GMI [15], they employ an architecture with only one branch. In this case, negative samples are obtained by corrupting the original graph. Different from the aforementioned *augmentation* schemes that generate congruent pairs to model *the joint distribution* of positive pairs, we resort to the term *corruption functions*, which approximate *the product of marginals*.

Stronger augmentation. Unlike GRACE [17] and GraphCL [16] that employ uniform edge/feature perturbation, GCA [18] proposes to perform adaptive augmentation based on importance scores of scores and feature dimensions. In this work, to involve less hyperparameters as possible, we focus on uniform transformation only.

Variants of contrasting modes. GMI [15] extends DGI [13] by further considering the agreement between raw node/edge features and node/edge representations. Because it requires much more computational resources, our experiments exclude this implementation of contrasting mode. In addition, there are several recent methods [27, 28] involve contrasting between local/global and *context* representations, which are usually derived from graph clustering algorithms. Considering the generality of the experiments, we shall leave it as a future direction.

Contrastive architectures without explicit negative samples. Recently, some GCL approaches relying on no negative samples have been proposed. BGRL [25] employs an asymmetric framework composing of online and offline encoders and directly use cosine similarity of two outputs as the self-supervision signal. GBT [26] proposes to construct a correlation matrix along the latent dimension instead of negative samples. In light of these methods, in this benchmarking study, we further conduct additional analysis on (1) the Bootstrapping Latent loss (BL), (2) Barlow Twins (BT) loss, and (3) VICReg loss.

3 Empirical Studies

The following section presents the empirical studies of graph contrastive learning. We comprehensively evaluate the model performance on different configurations. In the following section, we first introduce the experimental configurations and then summarize the results and observations regarding each particular component in the proposed paradigm. For details on the evaluational protocols and implementations, we refer readers of interest to Appendix B.

Table 2: Classification performance with different topology and feature augmentations. The best performance is highlighted in boldface and the second-to-best underlined. OOM indicates Out-Of-Memory on a 24GB GPU.

	Aug.	Node				Graph			
		Wiki	CS	Physics	Computers	PTC_MR	PROTEINS	REDDIT-B	IMDB-M
	None	71.73 ± 0.29	90.76 ± 0.05	93.69 ± 0.73	80.62 ± 0.62	50.22 ± 1.56	71.37 ± 0.48	68.64 ± 0.19	47.81 ± 0.14
Topo.	EA	79.42 ± 0.24	92.73 ± 0.10	94.77 ± 0.05	83.40 ± 0.64	58.34 ± 1.55	73.95 ± 0.90	79.40 ± 0.94	49.76 ± 0.64
	ER	78.61 ± 0.30	91.90 ± 0.06	95.21 ± 0.05	**87.84 ± 0.76**	**60.24 ± 2.27**	74.32 ± 0.42	83.30 ± 0.65	49.70 ± 0.48
	EF	75.70 ± 0.42	92.99 ± 0.15	94.88 ± 0.06	86.68 ± 0.73	58.22 ± 2.48	73.77 ± 0.42	83.69 ± 0.24	49.94 ± 0.21
	ND	79.57 ± 0.34	92.33 ± 0.12	**95.99 ± 0.12**	87.01 ± 0.54	59.77 ± 1.21	**74.96 ± 0.90**	**84.76 ± 0.87**	**50.30 ± 0.51**
	PPR	70.94 ± 0.43	92.25 ± 0.07	OOM	85.06 ± 0.53	56.08 ± 1.52	66.61 ± 0.90	70.66 ± 0.48	49.06 ± 0.37
	MKD	72.39 ± 0.49	92.62 ± 0.14	OOM	82.46 ± 0.58	56.66 ± 2.76	57.37 ± 0.34	71.87 ± 0.17	49.39 ± 0.25
	RWS	**79.74 ± 0.34**	**93.48 ± 0.08**	95.04 ± 0.11	87.60 ± 0.63	58.66 ± 2.39	72.13 ± 0.56	81.54 ± 0.78	49.96 ± 0.36
Feat.	FM	77.56 ± 0.46	91.55 ± 0.11	94.12 ± 0.21	**85.05 ± 0.51**	52.51 ± 2.43	**73.50 ± 0.64**	79.37 ± 0.03	49.47 ± 0.69
	FD	**77.74 ± 0.34**	**91.83 ± 0.08**	94.20 ± 0.16	84.93 ± 0.46	**53.69 ± 2.36**	72.95 ± 0.49	**80.94 ± 0.12**	**49.82 ± 0.23**

Datasets. We conduct experiments on a variety of datasets widely used in literature, ranging from academic networks to chemistry molecular datasets. For fair comparison, we closely follow previous studies on datasets preprocessing and experimental protocols [13, 14, 16–19, 25, 29–31].

Evaluation configurations. We mainly evaluate models with different design considerations on three benchmark tasks: (1) unsupervised node classification, (2) unsupervised graph classification, and (3) transfer learning. For all unsupervised tasks, we follow the linear evaluation scheme used by DGI [2] on nodes and InfoGraph [19] on graphs. The models are first trained in an unsupervised manner, and then the final embeddings are fed into a linear classifier to fit the labeled data. For transfer learning, we follow the settings in Hu et al. [31], where we first pretrain the encoder on part of the dataset and then finetune it on the rest. For all experiments, we run the model with ten random splits and report the averaged accuracies (%) as well as the standard deviation.

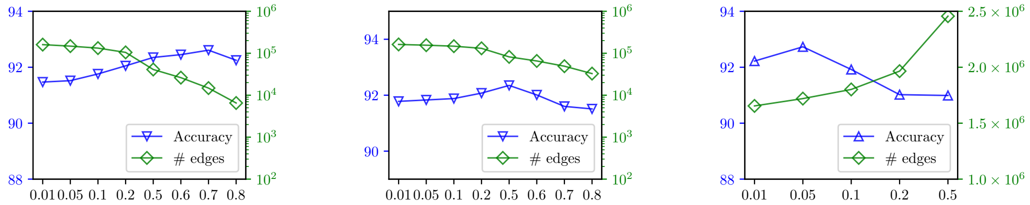
3.1 Data Augmentation

We first investigate how data augmentations affect the performance of GCL. Specifically, we apply different data augmentation functions to generate two views, leverage the InfoNCE objective, and contrast local-local representations. Except for augmentation functions used, all other settings are kept the same. The experimental results of employing different topology and feature augmentations are presented in Table 2. We further explore the use of compositional data augmentation schemes: (1) structure- and feature-level augmentations and (2) deterministic plus stochastic augmentations, where the results are summarized in Table 3.

Obs. 1. Topology augmentations greatly affect model performance. Augmentation functions that produce sparser graphs generally lead to better performance.

From Table 2, it is evident that the performance of GCL is highly dependent on the choice of topology augmentation functions. We observe that models that remove edges (ER, MDK, ND, PPR, and RWS), compared to models that add edges (EA), in general achieve better performance, which suggests that augmentation functions produce *sparser graph views* generally achieve better performance. We also find that RWS achieves better performance on node datasets, while ND favors graph tasks. We believe that by taking random walks we are able to better extract local structural patterns for one node. Since the graph datasets used in our study are of relatively small scales (< 500 nodes per graph), these random-walk-based sampling strategies may be confined and thus the simple node dropping (ND) scheme outperforms other augmentations on graph-level tasks.

To see how sparsity of the resulting views affect the performance, we further conduct sensitivity analysis on two models with ND, ER, and EA augmentations respectively, with varied dropping/adding probabilities on the CS dataset. The prediction accuracy along with the total number of edges in the produced graphs are plotted in Figure 2. From Figure 2a and 2b, we observe that model performance improves as more nodes are dropped and degenerates when the removal probability is too high. As seen in Figure 2c, the performance of EA augmentation downgrades greatly when more edges are added. In general, the result accords with our observations that many real-world graphs are inherently sparse [32, 33]. When too many edges are added, they connect nodes that are semantically unrelated, bringing noise to the generated graph view and thus deteriorate the quality of learned embeddings.



(a) Node dropping probability (b) Edge removing probability (c) Edge adding probability

Figure 2: Sensitivity analysis with varied topology augmentation probabilities on the CS dataset.

Table 3: Classification performance with different compositional augmentations.

(a) Joint augmentation on structure- and feature-level

Aug.	Node				Graph			
	Wiki	CS	Physics	Computers	PTC_MR	PROTEINS	REDDIT-B	IMDB-M
ER	78.61 ± 0.30	91.90 ± 0.06	95.21 ± 0.05	87.84 ± 0.76	60.24 ± 2.27	74.32 ± 0.42	83.30 ± 0.65	49.70 ± 0.48
+FM	79.09 ± 0.41	92.45 ± 0.10	95.95 ± 0.12	88.15 ± 0.53	**60.78 ± 1.52**	**75.47 ± 0.30**	83.67 ± 0.40	50.17 ± 0.39
+FD	**79.61 ± 0.38**	**93.63 ± 0.13**	**95.65 ± 0.06**	**88.42 ± 0.42**	53.88 ± 2.30	75.03 ± 0.23	**83.94 ± 0.60**	**50.30 ± 0.56**
ND	79.57 ± 0.34	92.33 ± 0.12	95.99 ± 0.12	87.01 ± 0.54	**59.77 ± 1.21**	74.96 ± 0.90	84.76 ± 0.87	**50.30 ± 0.51**
+FM	**79.80 ± 0.34**	92.61 ± 0.14	96.01 ± 0.05	87.17 ± 0.58	58.23 ± 0.99	**75.65 ± 0.27**	**84.94 ± 0.97**	49.82 ± 0.59
+FD	79.68 ± 0.40	**92.81 ± 0.07**	**96.15 ± 0.04**	**89.58 ± 0.40**	57.28 ± 2.33	75.21 ± 0.88	84.64 ± 0.34	50.18 ± 0.50
EA	79.42 ± 0.24	92.73 ± 0.10	94.77 ± 0.05	83.40 ± 0.64	58.34 ± 1.55	73.95 ± 0.90	79.40 ± 0.94	49.76 ± 0.64
+FM	79.69 ± 0.45	93.47 ± 0.07	**95.47 ± 0.05**	83.81 ± 0.91	**58.55 ± 1.99**	74.04 ± 0.40	79.98 ± 2.29	50.47 ± 0.38
+FD	**79.80 ± 0.17**	**93.62 ± 0.07**	94.94 ± 0.12	**84.63 ± 0.41**	55.11 ± 1.41	**74.89 ± 0.47**	**81.87 ± 1.86**	**50.55 ± 0.22**
RWS	79.74 ± 0.34	93.48 ± 0.08	95.04 ± 0.11	87.60 ± 0.63	58.66 ± 2.39	72.13 ± 0.56	81.54 ± 0.78	49.96 ± 0.36
+FM	**80.21 ± 0.30**	**94.35 ± 0.08**	**95.99 ± 0.08**	87.63 ± 0.31	**59.00 ± 1.88**	**74.88 ± 0.44**	81.14 ± 1.27	**50.74 ± 0.31**
+FD	79.78 ± 0.17	93.98 ± 0.09	95.66 ± 0.06	**89.38 ± 0.37**	56.85 ± 1.05	73.28 ± 0.38	**82.90 ± 1.33**	50.39 ± 0.30

(b) Composition of stochastic and deterministic augmentations

Aug.	Node				Graph			
	Wiki	CS	Physics	Computers	PTC_MR	PROTEINS	REDDIT-B	IMDB-M
PPR	70.94 ± 0.43	92.25 ± 0.07	OOM	85.06 ± 0.53	56.08 ± 1.52	66.61 ± 0.90	70.66 ± 0.48	49.06 ± 0.37
+ER	76.65 ± 0.42	92.41 ± 0.11	—	88.06 ± 0.62	**57.65 ± 2.48**	**73.76 ± 0.38**	**73.81 ± 0.48**	49.47 ± 0.24
+EA	77.71 ± 0.16	92.86 ± 0.16	—	86.29 ± 0.91	55.31 ± 1.80	68.55 ± 2.93	67.14 ± 4.82	**50.47 ± 0.31**
+ND	**78.16 ± 0.51**	92.77 ± 0.17	—	**89.11 ± 0.33**	56.19 ± 1.03	72.04 ± 1.96	72.40 ± 5.90	50.28 ± 0.27
+FD	74.12 ± 0.58	**93.55 ± 0.09**	—	86.49 ± 0.55	54.26 ± 1.71	72.51 ± 0.65	72.86 ± 0.30	49.28 ± 0.44
MKD	72.39 ± 0.49	92.62 ± 0.14	OOM	82.46 ± 0.58	56.66 ± 2.76	57.37 ± 0.34	71.87 ± 0.17	49.39 ± 0.25
+ER	**78.42 ± 0.40**	92.75 ± 0.05	—	89.60 ± 0.77	**60.97 ± 2.25**	71.51 ± 0.40	74.10 ± 0.45	49.63 ± 0.28
+EA	77.68 ± 0.55	92.71 ± 0.09	—	83.57 ± 1.08	54.60 ± 1.45	65.81 ± 3.01	65.65 ± 1.08	**50.61 ± 0.17**
+ND	77.35 ± 0.46	92.81 ± 0.05	—	**89.67 ± 0.48**	57.25 ± 1.87	72.40 ± 1.19	**76.65 ± 2.91**	50.41 ± 0.26
+FD	73.83 ± 0.42	**93.83 ± 0.14**	—	85.10 ± 0.92	54.46 ± 2.58	**72.76 ± 0.79**	73.82 ± 0.98	49.11 ± 0.68

Obs. 2. Feature augmentations bring additional benefits to GCL when discriminatory features are available.

From Table 2, we observe that the performance of models employing feature augmentations solely is inferior to that use topology augmentations. Also, we see that in general FD outperforms FM, which suggests the use of dropping features in an element-wise manner, though the performance increments are very limited.

Furthermore, we kindly note that graph datasets PTC_MR, RDT-B, and IMDB-M include no initial node features. Following previous work [19], we use 1-dimensional one's vector instead. It is seen that the performance gain of employing feature augmentations is not significant on these datasets that carry no informative node features. We thus posit that feature augmentation is helpful when discriminatory features are available. Along with Observation 1, we conclude that data augmentations in GCL should be adaptive to properties of the datasets, so the contrastive signal can better capture intrinsic patterns of graph structures and attributes, which corroborates the previous study [18].

Obs. 3. Compositional augmentations at both structure and attribute level benefit GCL most. Deterministic augmentation schemes should be used along with stochastic augmentations.

Table 4: Performance with different contrastive objectives and contrastive modes. L–L, G–L, and G–G denote local-local, global-local, and global-global contrasting modes. The best performing results for objectives (row-wise) and contrasting modes (column-wise) are highlighted in boldface and underline respectively.

(a) Node classification performance

Obj.	Wiki		CS		Physics		Computers	
	L–L	G–L	L–L	G–L	L–L	G–L	L–L	G–L
InfoNCE	**79.09 ± 0.15**	77.73 ± 0.94	**92.45 ± 0.83**	90.60 ± 0.06	**95.95 ± 0.92**	93.23 ± 0.96	**88.15 ± 0.59**	76.24 ± 0.93
JSD	78.83 ± 0.95	**78.71 ± 0.19**	92.18 ± 1.00	**91.31 ± 0.62**	94.32 ± 0.28	**94.12 ± 0.04**	82.02 ± 0.76	**78.27 ± 0.05**
TM	78.42 ± 0.88	76.53 ± 0.85	91.91 ± 0.31	90.11 ± 0.61	94.11 ± 0.60	92.78 ± 0.12	69.67 ± 0.88	76.38 ± 0.75

(b) Graph classification performance

Obj.	PTC-MR			PROTEINS			REDDIT-B			IMDB-M		
	L–L	G–L	G–G	L–L	G–L	G–G	L–L	G–L	G–G	L–L	G–L	G–G
InfoNCE	**61.21 ± 1.92**	60.21 ± 1.87	60.08 ± 1.14	**75.01 ± 0.25**	**74.92 ± 0.53**	**75.31 ± 0.21**	**83.12 ± 0.59**	82.62 ± 0.42	**83.45 ± 0.23**	**50.22 ± 0.21**	50.12 ± 0.25	49.23 ± 0.54
JSD	59.89 ± 2.39	**60.39 ± 1.73**	56.43 ± 2.51	74.11 ± 0.32	74.08 ± 0.40	74.32 ± 0.23	**82.66 ± 0.73**	81.98 ± 0.32	82.23 ± 0.47	50.02 ± 0.47	48.17 ± 0.28	49.34 ± 0.77
TM	54.92 ± 1.76	57.32 ± 2.31	**61.53 ± 0.72**	73.77 ± 0.47	71.31 ± 1.08	72.34 ± 1.16	79.32 ± 0.31	79.10 ± 0.83	83.21 ± 0.42	49.85 ± 0.69	48.40 ± 0.72	**50.87 ± 0.13**

From Table 3a, we observe that in most cases where meaningful node features are available, the use of feature augmentation in addition to structure augmentation benefits GCL, demonstrating that both topology and structures are important for learning graph representations. For datasets without informative node features, using feature augmentation may bring adverse effects, which corroborates our Observation 2. Also, we find that two feature augmentation schemes FM and FD perform similarly.

We also find that although two deterministic augmentation functions PPR and MDK do not perform well (cf. Table 2), joint utilization of stochastic and deterministic augmentation attains promising performance, echoing the design in MVGRL [14]. Recall that our contrastive objective is essentially aimed to discriminate between samples from the data distribution and samples from some "noise" distribution [34, 35]. Therefore, a stochastic augmentation scheme is needed to better approximate the noise distribution.

3.2 Contrasting Modes and Contrastive Objectives

The next experiments are concerned with how contrasting modes and contrastive objectives affect the model performance. We train the model with different contrasting modes and contrastive objectives, with topology augmentation set to ER and feature augmentation to FM. We also conduct experiments based on ND and FD, and we see no clear difference than that on ER and FM. Except that embedding sizes are fixed, other hyperparameters are tuned to obtain the best performance under each experiment for fair comparison. Table 4a and 4b present the experimental results on unsupervised classification tasks, and Table 5 presents performance on transfer learning (trained with the InfoNCE objective).

Obs. 4. Downstream tasks of different granularities favor different contrasting modes.

What stands out from the table is that contrasting local-local pairs achieves the best performance on node-level classification, while global-global and global-local modes perform better on graph-level tasks. Also, from Table 5 we observe similar trends on transfer learning of graph classification, where local-global and global-global contrasting modes generally perform better. This suggests us to choose a contrasting mode corresponding to granularity of the end task, i.e. local-local for node-level tasks and global-local or global-global for graph-level tasks.

For graph-level tasks, contrasting the global-local embedding pairs explicitly encodes global information into global representations. However, its performance on node-level tasks is inferior to that uses

Table 5: Downstream performance on transfer learning of graph classification.

Mode	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	PPI
L–L	72.26 ± 0.81	74.79 ± 0.52	62.33 ± 0.53	60.43 ± 1.35	75.11 ± 2.16	71.41 ± 1.97	77.96 ± 0.72	**78.53 ± 1.84**	65.53 ± 0.58
G–L	**73.12 ± 0.62**	**74.89 ± 0.43**	**62.45 ± 0.67**	**60.62 ± 1.08**	75.01 ± 2.37	**71.62 ± 2.02**	78.02 ± 0.84	77.43 ± 1.28	66.46 ± 0.98
G–G	71.52 ± 0.57	73.32 ± 0.78	62.34 ± 0.66	60.40 ± 0.99	**76.03 ± 2.33**	70.24 ± 1.93	**78.24 ± 0.99**	76.23 ± 1.48	**67.23 ± 0.65**

local-local mode. We may explain the results from the perspective of the optimization objective. The global-local mode is essentially a proxy for local-local CL, provided that the readout function r is expressive enough [36, 13]. However, in reality, the injectivity property of the readout function is hard to fulfill [37]. Therefore, the readout function may not distill enough information from node-level embeddings, leading to performance gap between node- and graph-level tasks.

In accordance with the presented results, recent studies [38, 39] have made a similar finding for learning visual representations. They demonstrate that being trained on instance-level pretext tasks (i.e. contrasting image-level embeddings in the same batch), current CL models have suboptimal performance in fine-grained tasks, e.g., semantic segment that requires pixel-level details.

Obs. 5. The use of InfoNCE objective leads to consistent improvements across all settings yet requires a rather large number of negative samples.

From Table 4a, it is seen that the InfoNCE achieve the best performance among contrastive objectives, which has already been shown effective by many recent methods [4, 7, 8, 40, 41].

Among the studied three objectives, recent studies have already revealed that the InfoNCE loss has an intrinsic ability to perform hard negative samples [23]. Particularly, one very recent study in computer vision [42] shows that the use of a temperature parameter τ in the InfoNCE objective acts as an adjustment factor to exert penalties on hard negative samples. To verify this on GCL, we further conduct sensitivity analysis on this temperature parameter on both node (CS) and graph (IMDB-M) datasets as shown in Figure 3. We observe that with the increase of τ, the performance improves at first and downgrades later, with not much performance fluctuation. According to Wang and Liu [42], the InfoNCE objective pays less attention to hard negatives as τ increases. This hardness-aware behavior demonstrates the importance of *striking a balance* between separation of hardest negative samples ($\tau \to 0^+$) and global uniformity ($\tau \to \infty$) on GCL.

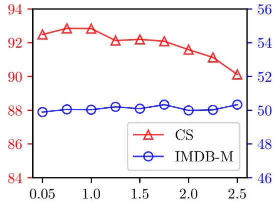


Figure 3: Sensitivity analysis of τ in the InfoNCE objective.

We also note that we obtain close-to-optimal performance on several graph datasets when we train the model using the triplet loss to contrast global-global embeddings. This could be explained by the fact that the triplet loss is known to be sensitive to hard negatives [43]. In the global-global mode, the contrasting pairs are graph embeddings in the batch, which are more distinct (harder) than each other, compared to node pairs. Therefore, we suspect that it provides more informative signals to hard negative samples that boosts model learning.

3.3 Negative Mining Strategies

Following the previous section, we probe the explicit use of negative mining strategies on top of contrastive objectives, which essentially measure the hardness of each negative pair and upweight hard negative samples. We train four models on three node classification datasets (Wiki, CS, and Computers) using the local-local mode with the InfoNCE objective. The results are presented in Figure 4.

Obs. 6. Existing negative mining techniques bring limited benefit to GCL.

From the figure, we see that the mixup strategy (HNM) consistently improves the performance on all three datasets, though the improvements are marginal. With more hard negatives being synthesized, the contrastive objective gives larger weights to harder negatives (i.e. embeddings close to the anchor), which improves the discriminative power of the contrastive model. The other three strategies, while slight improvement can be observed under certain hyperparameter configurations, do not in general improve the InfoNCE baseline by much.

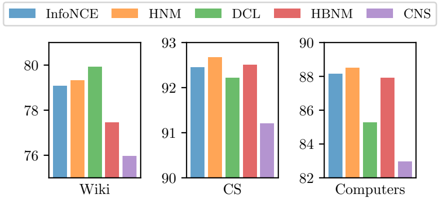


Figure 4: Performance comparison with different negative mining strategies.

In existing formulation of these four negative mining techniques, the sample hardness is measured by inner product of sample embeddings. Since we are in a fully unsupervised setting, no label (class) information can be accessed. Under existing contrasting modes, for one anchor sample, the contrastive objective pushes all different representations away, *irrespective of their real semantic relations*. What is even worse, most GNN models tend to produce similar embeddings for neighboring nodes regardless of semantic classes [44–47], which may further bias the selection of hard negatives. Therefore, we argue that there is disagreement between semantic similarity and example hardness. By selecting hard negative samples merely according to similarity measure of embeddings, these hard negatives are potentially *positive samples*, which produces adverse learning signals to the contrastive objective. We kindly find that our discovery reminiscent to one very recent study in visual contrastive learning [42], which recommends an adaptive scheduling scheme for the temperature parameter when using InfoNCE as the contrastive objective, so that hard but false negatives could be tolerated as the training progresses.

3.4 Summary of Additional Experiments

We conduct additional experiments regarding large-scale evaluation and negative-sample-free contrastive objectives. Due to space limitation, we summarize our key findings here and readers of interest may refer to Appendix C for details.

Large-scale evaluation. Current CL models suffer from large computational burden due to the need of negative samples. Since the experiments are limited to small- to medium-scale datasets, we conduct a further study assessing how existing GCL work scales to large-scale graphs using two open graph benchmark [48] datasets. From the experimental results, we find that our main conclusions are still applicable to large-scale datasets.

Experiments on negative-sample-free contrastive objectives Besides contrastive objectives that rely on negative samples, we experiment with three negative-sample-free objectives: Bootstrapping Latent (BL) loss, Barlow Twins (BT) loss, and VICReg loss. We observe from the results that these three objectives greatly reduce the computational burden since no explicit negative samples are constructed. More surprisingly, Barlow Twins and VICReg losses are even able to achieve better performance compared to their negative-sample-based counterparts.

3.5 Discussions on Limitations and Future Directions

Due to limited space, some limitations of our work need to be acknowledged.

- **Limited design considerations.** In this work, we consider limited design considerations, namely four design dimensions. An issue that is not addressed in this study is what role do many other model-specific factors, e.g., whether to employ a projection head in the InfoNCE objective and what graph encoders should be employed, play in GCL.
- **Limited downstream tasks.** Our empirical study only includes experiments on node- and graph-level classification and graph-level transfer learning; a boarder range of end tasks of different granularities, e.g., link prediction and community detection, may be more beneficial to draw convincing conclusions.
- **Lack of theoretical justification.** Our work only presents empirical studies which has thrown up many questions in need of further theoretical justification for better understanding the mechanisms underlying GCL, for instance, how to appropriately measure and select hard negative samples for contrastive objectives in the graph domain.

Our empirical findings also suggest several future directions that may be helpful for improving GCL.

- **Towards automated augmentation.** We understand that topology augmentation is of paramount importance to GCL. However, existing work leverages manually designed ad-hoc augmentation strategies, which may result in suboptimal performance. Recent studies in graph structure learning establish a principled way to learn optimal structures of graph-structured data [33], which we argue could be used for automatically learn augmentation functions suitable for GCL tasks.
- **Understanding the performance gap between pretext and downstream tasks.** We empirically demonstrate the correlation between the choice of end tasks and contrastive objectives, yet calls for a thorough understanding for the performance gap between pretext and downstream tasks. We have found that there is some progress in this regard [49], but it is far from fully explored.
- **Structure-aware negative sampling.** Unlike in computer vision fields, similar visual features may naturally correlate to closer semantic categories, measuring the hardness through embedding

similarities in graph-structured data is more difficult. A series of earlier work in network embedding proposes solutions from structural aspects [50–52]. However, how to integrate rich structure information for modeling better negative distributions for GCL is still left unexplored.

4 Conclusion

In this paper, we first present a taxonomy for GCL, where we categorize existing work to four aspects: data augmentations, contrasting modes, contrastive objective, and negative sampling techniques. Then, we analyze the choices in the design space for each aspect by extensively study the empirical performance of models under different design choices over a comprehensive set of benchmarking tasks and datasets. Our rigorous analysis results in interesting findings about the interplay of design dimensions in GCL. We also provide a handy toolbox `PyGCL` to facilitate the implementation and experimenting of GCL models. We hope our empirical study provides several practical guidelines for future research in this vigorous field.

References

[1] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.

[2] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, 2018.

[3] 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.

[4] Philip Bachman, R. Devon Hjelm, and William Buchwalter. Learning Representations by Maximizing Mutual Information Across Views. In *NeurIPS*, pages 15509–15519, 2019.

[5] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*, pages 9726–9735, 2020.

[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *NeurIPS*, pages 9912–9924, 2020.

[7] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning. In *NeurIPS*, pages 21271–21284, 2020.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, pages 1597–1607, 2020.

[9] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge J. Belongie, and Yin Cui. Spatiotemporal Contrastive Video Representation Learning. In *CVPR*, 2021.

[10] Tassilo Klein and Moin Nabi. Contrastive Self-Supervised Learning for Commonsense Reasoning. In *ACL*, pages 7517–7523, 2020.

[11] Hongchao Fang, Sicheng Wang, Meng Zhou, Jiayuan Ding, and Pengtao Xie. CERT: Contrastive Self-supervised Learning for Language Understanding. *arXiv.org*, 2020.

[12] Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. *arXiv.org*, April 2021.

[13] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep Graph Infomax. In *ICLR*, 2019.

[14] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive Multi-View Representation Learning on Graphs. In *ICML*, pages 3451–3461, 2020.

[15] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph Representation Learning via Graphical Mutual Information Maximization. In *WWW*, pages 259–270, 2020.

[16] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph Contrastive Learning with Augmentations. In *NeurIPS*, pages 5812–5823, 2020.

[17] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep Graph Contrastive Representation Learning. In *GRL+@ICML*, 2020.

[18] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph Contrastive Learning with Adaptive Augmentation. In *WWW*, pages 2069–2080, 2021.

[19] 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.

[20] Yaochen Xie, Zhao Xu, Zhengyang Wang, and Shuiwang Ji. Self-Supervised Learning of Graph Neural Networks: A Unified Review. *arXiv.org*, February 2021.

[21] Yixin Liu, Shirui Pan, Ming Jin, Chuan Zhou, Feng Xia, and Philip S. Yu. Graph Self-Supervised Learning: A Survey. *arXiv.org*, February 2021.

[22] Lirong Wu, Haitao Lin, Zhangyang Gao, Cheng Tan, and Stan Z. Li. Self-supervised on Graphs: Contrastive, Generative, or Predictive. *arXiv.org*, May 2021.

[23] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised Contrastive Learning. In *NeurIPS*, pages 18661–18673, 2020.

[24] 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 *KDD*, pages 1150–1160, 2020.

[25] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. Bootstrapped Representation Learning on Graphs. *arXiv.org*, February 2021.

[26] Piotr Bielak, Tomasz Kajdanowicz, and Nitesh V. Chawla. Graph Barlow Twins: A Self-Supervised Representation Learning Framework for Graphs. *arXiv.org*, June 2021.

[27] Shichang Zhang, Ziniu Hu, Arjun Subramonian, and Yizhou Sun. Motif-Driven Contrastive Learning of Graph Representations. *arXiv.org*, December 2020.

[28] Costas Mavromatis and George Karypis. Graph InfoClust: Maximizing Coarse-Grain Mutual Information in Graphs. In *PAKDD*, pages 541–553, 2021.

[29] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of Graph Neural Network Evaluation. In *R2L@NeurIPS*, 2018.

[30] Marinka Zitnik, Rok Sosič, Marcus W. Feldman, and Jure Leskovec. Evolution of Resilience in Protein Interactomes Across the Tree of Life. *PNAS*, 116(10):4426–4433, March 2019.

[31] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for Pre-training Graph Neural Networks. In *ICLR*, 2020.

[32] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph Structure Learning for Robust Graph Neural Networks. In *KDD*, pages 66–74. ACM, 2020.

[33] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep Graph Structure Learning for Robust Representations: A Survey. *arXiv.org*, March 2021.

[34] Michael Gutmann and Aapo Hyvärinen. Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. *J. Mach. Learn. Res.*, 13:307–361, 2012.

[35] Andriy Mnih and Koray Kavukcuoglu. Learning Word Embeddings Efficiently with Noise-Contrastive Estimation. In *NIPS*, pages 2265–2273, 2013.

[36] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *ICLR*, 2019.

[37] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan R. Salakhutdinov, and Alexander J. Smola. Deep Sets. In *NIPS*, pages 3391–3401, 2017.

[38] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu. Propagate Yourself: Exploring Pixel-Level Consistency for Unsupervised Visual Representation Learning. In *CVPR*, 2021.

[39] Elijah Cole, Xuan Yang, Kimberly Wilber, Oisin Mac Aodha, and Serge Belongie. When Does Contrastive Visual Representation Learning Work? *arXiv.org*, May 2021.

[40] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive Multiview Coding. In *ECCV*, pages 776–794, 2020.

[41] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What Makes for Good Views for Contrastive Learning? In *NeurIPS*, pages 6827–6839, 2020.

[42] Feng Wang and Huaping Liu. Understanding the Behaviour of Contrastive Loss. In *CVPR*, 2021.

[43] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *CVPR*, pages 815–823, 2015.

[44] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*, pages 3538–3545, 2018.

[45] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. In *AAAI*, pages 3438–3445, 2020.

[46] Yanqiao Zhu, Weizhi Xu, Qiang Liu, and Shu Wu. When Contrastive Learning Meets Active Learning: A Novel Graph Active Learning Paradigm with Self-Supervision. *arXiv.org*, October 2020.

[47] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two Sides of the Same Coin: Heterophily and Oversmoothing in Graph Convolutional Neural Networks. *arXiv.org*, February 2021.

[48] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS*, pages 22118–22133, 2020.

[49] Qi Zhu, Yidan Xu, Haonan Wang, Chao Zhang, Jiawei Han, and Carl Yang. Transfer Learning of Graph Neural Networks with Ego-graph Information Maximization. *arXiv.org*, September 2020.

[50] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. In *AAAI*, pages 2508–2515, 2018.

[51] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD*, pages 974–983. ACM, 2018.

[52] Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. Understanding Negative Sampling in Graph Representation Learning. In *KDD*, pages 1666–1676. ACM, 2020.

[53] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric. In *RLGM@ICLR*, 2019.

[54] Péter Mernyei and Catalina Cangea. Wiki-CS: A Wikipedia-Based Benchmark for Graph Neural Networks. In *GRL+@ICML*, July 2020.

[55] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking Graph Neural Networks. *arXiv.org*, March 2020.

[56] Pinar Yanardag and S. V. N. Vishwanathan. Deep Graph Kernels. In *KDD*, pages 1365–1374, 2015.

[57] Teague Sterling and John J Irwin. ZINC 15: Ligand Discovery for Everyone. *J. Chem. Inf. Model.*, 55(11):2324–2337, 2015.

[58] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay S. Pande. MoleculeNet: A Benchmark for Molecular Machine Learning. *Chem. Sci.*, 9:513–530, 2018.

[59] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, pages 8024–8035, 2019.

[60] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. *TIST*, 2(3):1–27, April 2011.

[61] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs. *arXiv.org*, March 2021.

[62] Pierre H. Richemond, Jean-Bastien Grill, Florent Altché, Corentin Tallec, Florian Strub, Andrew Brock, Samuel Smith, Soham De, Razvan Pascanu, Bilal Piot, and Michal Valko. BYOL Works Even Without Batch Statistics. *arXiv.org*, October 2020.

[63] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[64] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *CVPR*, pages 1–9, 2015.

[65] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Colorization as a Proxy Task for Visual Understanding. In *CVPR*, pages 840–849, 2017.

[66] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised Representation Learning by Predicting Image Rotations. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[67] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised Feature Learning via Non-Parametric Instance Discrimination. In *CVPR*, pages 3733–3742, 2018.

[68] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. Fast Random Walk with Restart and Its Applications. In *ICDM*, pages 613–622, 2006.

[69] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab, November 1999.

[70] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion Improves Graph Learning. In *NeurIPS*, pages 13333–13345, 2019.

[71] François Fouss, Kevin Françoisse, Luh Yen, Alain Pirotte, and Marco Saerens. An Experimental Investigation of Kernels on Graphs for Collaborative Recommendation and Semisupervised Classification. *Neural Networks*, 31:53–72, 2012.

[72] Hao Zhu and Piotr Koniusz. Simple Spectral Graph Convolution. In *ICLR*, 2021.

[73] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks From Overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[74] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv.org*, 2018.

[75] Ralph Linsker. Self-Organization in a Perceptual Network. *IEEE Computer*, 21(3):105–117, 1988.

[76] Michael Gutmann and Aapo Hyvärinen. Noise-Contrastive Estimation: a New Estimation Principle for Unnormalized Statistical Models. In *AISTATS*, pages 297–304, 2010.

[77] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *NIPS*, pages 271–279, 2016.

[78] Ben Poole, Sherjil Ozair, Aäron van den Oord, Alexander A. Alemi, and George Tucker. On Variational Bounds of Mutual Information. In *ICML*, pages 5171–5180, 2019.

[79] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. On Mutual Information Maximization for Representation Learning. In *ICLR*, 2020.

[80] Tongzhou Wang and Phillip Isola. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *ICML*, pages 9929–9939, 2020.

[81] Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding Self-Supervised Learning Dynamics without Contrastive Pairs. In *ICML*, 2021.

[82] Zekarias T. Kefato and Sarunas Girdzijauskas. Self-Supervised Graph Neural Networks Without Explicit Negative Sampling. In *SSL@WWW*, 2021.

[83] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, pages 448–456, 2015.

[84] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. In *ICML*, 2021.

[85] Yao-Hung Hubert Tsai, Shaojie Bai, Louis-Philippe Morency, and Ruslan R. Salakhutdinov. A Note on Connecting Barlow Twins with Negative-Sample-Free Contrastive Learning. *arXiv.org*, April 2021.

[86] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning. *arXiv.org*, May 2021.

[87] Naftali Tishby, Fernando C. Pereira, and William Bialek. The Information Bottleneck Method. *arXiv.org*, April 2000.

[88] Naftali Tishby and Noga Zaslavsky. Deep Learning and the Information Bottleneck Principle. In *ITW*, pages 1–5, 2015.

[89] Jovana Mitrovic, Brian McWilliams, and Melanie Rey. Less Can Be More in Contrastive Learning. In *ICBINB@NeurIPS*, pages 70–75, 2020.

[90] Tiffany Tianhui Cai, Jonathan Frankle, David J. Schwab, and Ari S Morcos. Are All Negatives Created Equal in Contrastive Instance Discrimination? *arXiv.org*, October 2020.

[91] Ching-Yao Chuang, Joshua Robinson, Lin Yen-Chen, Antonio Torralba, and Stefanie Jegelka. Debiased Contrastive Learning. In *NeurIPS*, pages 8765–8775, 2020.

[92] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard Negative Mixing for Contrastive Learning. In *NeurIPS*, pages 21798–21809, 2020.

[93] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive Learning with Hard Negative Samples. In *ICLR*, 2021.

[94] Mike Wu, Milan Mosse, Chengxu Zhuang, Daniel Yamins, and Noah Goodman. Conditional Negative Sampling for Contrastive Learning of Visual Representations. In *ICLR*, 2021.

[95] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-Mix: A Strategy for Regularizing Contrastive Representation Learning. *arXiv.org*, October 2020.

[96] Vikas Verma, Minh-Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc V. Le. Towards Domain-Agnostic Contrastive Learning. *arXiv.org*, November 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Section 3.4
   (c) Did you discuss any potential negative societal impacts of your work? [No] We are not aware of potential negative societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Appendix B
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Appendix B
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A] No new assets are used. All datasets can be publicly accessed.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Reproducibility of Experiments

A.1 Brief Introduction of PyGCL

PyGCL is a battery-included toolkit for implementing graph contrastive learning models, and it is extensively used in our work to implement and execute all our experiments. PyGCL implements four main components of graph contrastive learning algorithms, which strictly follows our proposed design dimensions in this work:

- Graph augmentation: transforms input graphs into congruent graph views.
- Contrasting modes: specifies positive and negative pairs.
- Contrastive objectives: computes the likelihood score for positive and negative pairs.
- Negative mining strategies: improves the negative sample set by considering hardness of negative sample.

PyGCL also implements utilities for loading datasets, training models, and running experiments.

Graph augmentations. In `GCL.augmentors`, PyGCL provides the `Augmentor` base class, which offers a universal interface for graph augmentation functions. A list of graph augmentation functions and the class name of its implementation in PyGCL in given in Table 6. Due to complexity issues, Edge Flipping (EF) is implemented as a composition of Edge Adding and Edge Removing in PyGCL. Just as what its name indicates, the `Identity` augmentation will return the graph as it is.

Table 6: Graph augmentations and its corresponding class in PyGCL.

Augmentation	Class name
Edge Adding (EA)	`EdgeAdding`
Edge Removing (ER)	`EdgeRemoving`
Feature Masking (FM)	`FeatureMasking`
Feature Dropout (FD)	`FeatureDropout`
Personalized PageRank (PPR)	`PPRDiffusion`
Markov Diffusion Kernel (MDK)	`MarkovDiffusion`
Node Dropping (ND)	`NodeDropping`
Subgraphs induced by Random Walks (RWS)	`RWSampling`
Ego-net Sampling (ES)	`Identity`

Contrasting modes. PyGCL implements three contrasting modes described in our paper, including local-local, global-local, and global-global CL. Note that the bootstrapping latent loss involves some special model design (asymmetric online/offline encoders and momentum weight updates) and thus we implement contrasting modes involving this contrastive objective in a separate `BGRL` model.

Contrastive objectives. In `GCL.losses`, PyGCL includes the following contrastive objectives: InfoNCE, Jason-Shannon Divergence (JSD), Triplet Margin (TM), Bootstrapping Latent (BL), Barlow Twins (BT), and VICReg losses, shown in Table 7.

Table 7: Contrastive objectives and its corresponding class in PyGCL.

Augmentation	Class name
InfoNCE loss	`InfoNCELoss`
Jensen-Shannon Divergence (JSD) loss	`JSDLoss`
Triplet Margin (TM) loss	`TripletLoss`
Bootstrapping Latent (BL) loss	`BootstrapLoss`
Barlow Twins (BT) loss	`BTLoss`
VICReg loss	`VICRegLoss`

All these objectives are for contrasting positive and negative pairs at the same scale (i.e. local-local and global-global modes). For global-local modes, we offer `G2L` variants except for Barlow Twins and VICReg losses. Moreover, for InfoNCE, JSD, and Triplet losses, we further provide `G2LEN` variants, primarily for node-level tasks, which involve explicit construction of negative samples.

Hard negative mining strategies. In `GCL.losses`, PyGCL further implements four hard negative mining strategies: hard negative mixing, conditional negative sampling, debiased contrastive objective, and hardness-aware negative sampling, as summarized in Table 8. All these strategies are build upon the InfoNCE contrastive objective.

Table 8: Hard negative mining strategies and its corresponding implementations in PyGCL.

Augmentation	Class name
Hard negative mixing	`HardMixingLoss`
Conditional negative sampling	`RingLoss`
Debiased contrastive objective	`InfoNCELoss(debiased_nt_xent_loss)`
Hardness-biased negative sampling	`InfoNCELoss(hardness_nt_xent_loss)`

Utilities for experiment management. PyGCL provides various tools for data loading, model training, and experiment execution. Based on PyTorch Geometric [53], it provides functions for loading common node and graph datasets. It also implements the training of GCL models with different contrasting modes and contrastive objectives. Furthermore, it provides tool experiments for running experiments with different hyperparameter configurations in parallel on multiple GPU instances.

A.2 Reproducing Results in Our Work.

We briefly list instructions for reproducing the experiments in this paper.

- Experiments with data augmentations (Table 2 and 3) can be reproduced by executing the training scripts `train_node_l2l.py` and `train_graph_l2l.py` and changing augmentation parameters as specified in the file passed to the `-param_path` argument.
- Experiments with different contrasting modes and contrastive objectives (Table 4a and 4b) can be reproduced by the following steps:
  - For node-level tasks, `train_node_l2l.py` runs local-local CL and `train_node_g2l.py` runs local-global CL.
  - For graph-level tasks, `train_graph_l2l.py` runs local-local CL, `train_graph_g2l.py` global-local CL, and `train_graph_g2g.py` global-global CL.
  - Different contrastive losses can be specified by passing the loss name to `loss` argument.
  - Specially, the results of local-local and global-local with Bootstrapped Latent (BL) loss can be obtained using `train_node_BGRL_l2l.py` and `train_node_BGRL_g2l.py` respectively, and results of local-local, global-local, and global-global with Bootstrapped Latent (BL) loss can be obtained using `train_graph_BGRL_l2l.py`, `train_graph_BGRL_g2l.py`, and `train_graph_g2g.py` respectively.
- Model performance of transfer learning in Table 5 can be reproduced following instructions in the `transfer_learning` directory.

B Experimental Protocols

Datasets. We conduct experiments on a variety of datasets widely used in literature, ranging from academic networks to chemistry molecular datasets. The statistics of all used datasets are summarized in Table 9 and 10 respectively.

For unsupervised classification datasets, we use a total of eight datasets, including Wiki [54], Computers, CS, and Physics [55] for node classification and PTC_MR, PROTEINS, REDDIT-B, and IMDB-M [56] for graph classification. For transfer learning, we use two pretraining datasets for molecular property prediction in chemistry and protein function prediction in biology respectively. In the chemistry domain, we use the ZINC15 dataset [57] for pretraining and eight binary graph classification datasets in MoleculeNet [58] for downstream evaluation. For the biology domain, we use a 395K PPI network of 50 species for pretraining and another PPI network consisting of 88K proteins from 8 different species [30] for finetuning.

Implementation details. We choose GCN [1] as the encoder for all node tasks, and use GIN [36] for all graph tasks. In different experiments, we select embedding dimension among [64, 128, 256, 512], learning rate among [0.0001, 0.001, 0.01, 0.1], the number of GNN layers among [2, 3, 4], and weight decay among $[10^{-5}, 10^{-6}, 10^{-7}, 5 \times 10^{-8}, 10^{-8}]$. The temperature τ in the InfoNCE loss is

Table 9: Statistics of datasets used for unsupervised learning experiments.

Dataset	Domain	Task	#Graphs	Avg. #nodes	Avg. #edges	#Features	#Classes
Wiki	Knowledge base	Unsupervised node classification	1	11,701	216,123	300	10
Computers	Social networks		1	13,752	245,861	767	10
CS	Citation networks		1	18,333	81,894	6,805	15
Physics	Citation networks		1	34,493	247,962	8,415	5
PTC_MR	Biochemical molecules	Unsupervised graph classification	344	14.29	14.69	—	2
PROTEINS	Biochemical molecules		1,133	39.06	145.63	18	6
REDDIT-B	Social networks		2,000	429.63	497.75	—	2
IMDB-M	Social networks		1,500	13.00	65.94	—	3

Table 10: Statistics of datasets used for transfer learning experiments.

Datasets	Category	Utilization	#Graphs	Avg. #nodes	Avg. degree
ZINC-2M	Biochemical molecules	Pretraining	2,000,000	26.62	57.72
PPI-306K	Protein-protein intersection networks		306,925	39.82	729.62
BBBP	Biochemical molecules	Finetuning	2,039	24.06	51.9
Tox21	Biochemical molecules		7,831	18.57	38.58
ToxCast	Biochemical molecules		8,576	18.78	38.52
SIDER	Biochemical molecules		1,427	33.64	70.71
ClinTox	Biochemical molecules		1,477	26.15	55.76
MUV	Biochemical molecules		93,087	24.23	52.55
HIV	Biochemical molecules		41,127	25.51	54.93
BACE	Biochemical molecules		1,513	34.08	73.71
PPI	Protein-protein intersection networks		88,000	49.35	890.77

chosen from 0.1 to 0.9, and the batch size for graph datasets is chosen between [32, 64, 128, 256, 512]. Early stopping strategy is applied with a window size of 50, and the epoch with the smallest loss will be used in evaluation. All experiments are run on GeForce RTX 3090 GPUs with 24GB memory, and all models are written with PyTorch 1.8.1 [59] and PyTorch Geometric 1.7.0 [53] on Python 3.8.8.

Evaluation protocols. We mainly evaluate models with different design considerations on three benchmark tasks: (1) unsupervised node classification, (2) unsupervised graph classification, and (3) transfer learning. For all unsupervised tasks on nodes, we follow the linear evaluation scheme used by [2], where the models are first trained in an unsupervised manner, and then the final embeddings is fed into a ℓ_2-regularized logistic regression classifier to fit the labeled data. For graph classification tasks, we follow InfoGraph [19], where we train the graph encoders at first and evaluate the embeddings by training another SVM model [60]. The C parameter is selected from a range of $[10^{-3}, 10^{-2}, \ldots, 10^{3}]$. For Wiki For transfer learning, we follow the settings in Hu et al. [31], where we first pretrain the encoder on part of the dataset, then finetune it on the other dataset. Then, For all experiments, we run the model with ten random splits and report the averaged accuracies (%) as well as the standard deviation.

C Additional Experiments

C.1 Large-Scale Evaluation

Besides standard classification tasks and transfer learning tasks, we further evaluate existing GCL work on large-scale datasets from the Open Graph Benchmark [48, 61].

Summary of datasets, tasks, and metrics. We use two additional datasets ogbg-molhiv for binary graph classification and a downsampled version of PCQM4M-LSC for graph regression. In the two datasets, each graph represents a molecule, where nodes and edges correspond to atoms and chemical bonds, respectively. Since there are edge features associated with each graph, we use the GINE network proposed in Hu et al. [31] as the encoder. For edge features, we also examine two augmentation schemes Edge Attribute Masking (EAM) and Edge Attribute Dropout (EAD), similar to node-level FM and FD (denoted as NFM and NFD). All other evaluation protocols remain the same as previous. Details of statistics of the two datasets are summarized in Table 11.

Table 11: Summary of large-scale datasets.

Dataset	Domain	Task	#Graphs	Avg. #nodes	Avg. #edges	#Features	Metric
ogbg-molhiv	Molecules	Binary graph classification	41,127	25.5	27.5	9 (nodes) 3 (edges)	ROC-AUC
PCQM4M-10K		Graph regression	10,000	14.1	29.1		MAE

Table 12: Performance (ogbg-molhiv in ROC-AUC and PCQM4M-10K in MAE) with different augmentation schemes.

Augmentation	ogbg-molhiv	PCQM4M-10K
ER	**67.97 ± 3.66**	0.5569 ± 0.0194
ND	66.92 ± 4.05	**0.5532 ± 0.1064**
RWS	64.86 ± 2.90	0.5697 ± 0.1336
NFM	53.60 ± 1.76	0.6387 ± 0.1302
NFD	54.82 ± 1.67	0.6026 ± 0.1640
EAM	53.60 ± 1.76	0.6387 ± 0.1302
EAD	54.82 ± 1.67	0.6026 ± 0.1640

For ogbg-molhiv, the task is to predict a certain molecular property, measured in terms of Receiver Operating Characteristic Area Under Curve (ROC-AUC) scores. We follow the official scaffold splitting where structurally different molecules are separated into different subsets.

For PCQM4M-LSC, we randomly subsample 10K graphs according to PubChem ID (CID) and denote the resulting dataset as PCQM4M-10K. The regression task is to predict HOMO-LUMO energy gap in electronvolt (eV) given 2D molecular graphs. We report performance in terms of Mean Absolute Error (MAE).

Experiments on different augmentation schemes. In Table 12, we report the performance on the two large scale datasets with different data augmentations while keeping the contrasting mode to global-global and the objective to InfoNCE. We have findings consistent with those in Observation 1 that (1) both ER and ND have competitive performance consistently on the two datasets; (2) RWS is inferior to the other two topology augmentation scheme due to the limited size of graphs, as pointed out in Observation 1; and (3) solely performing feature augmentations (NFM, NFD, EAM, and EAD) can not bring satisfying performance, because structural information plays more important role in GCL.

Experiments on contrasting modes and contrastive objectives. Then, we examine different contrasting modes and contrastive objectives, where the results are shown in Table 13. Data augmentations are set to the combination of ER and FM in all variants. From the table, we observe trends consistent to those in Table 4b and 5 that the global-local and global-global modes yield competitive performance on graph tasks, and InfoNCE outperforms other objectives under most settings.

C.2 Experiments on Negative-Sample-Free Contrastive Objectives

Besides contrastive objectives that rely on negative samples, we experiment with three negative-sample-free objectives: Bootstrapping Latent (BL) loss, Barlow Twins (BT) loss, and VICReg loss.

Table 13: Performance (ogbg-molhiv in ROC-AUC and PCQM4M-10K in MAE) with different contrasting modes and contrastive objectives. The best performing results for objectives (row-wise) and contrasting modes (column-wise) are highlighted in boldface and underline respectively.

Obj.	ogbg-molhiv			PCQM4M-10K		
	L–L	G–L	G–G	L–L	G–L	G–G
InfoNCE	**63.22 ± 2.92**	58.26 ± 3.92	**65.48 ± 4.43**	**0.5621 ± 0.1911**	0.5640 ± 0.0081	**0.5464 ± 0.1689**
JSD	60.02 ± 2.98	**60.98 ± 2.42**	59.51 ± 2.01	0.5877 ± 0.0173	**0.5216 ± 0.0221**	0.5925 ± 0.1441
TM	58.42 ± 6.40	57.97 ± 7.57	58.48 ± 0.65	0.6315 ± 0.1881	0.6156 ± 0.0749	0.5743 ± 0.0110

Table 14: Performance with negative-sample-free contrastive architectures. L–L, G–L, and G–G denote local-local, global-local, and global-global contrasting modes. The best performing results for objectives (row-wise) and contrasting modes (column-wise) are highlighted in boldface and underline respectively.

(a) Node classification performance

Obj.	Wiki		CS		Physics		Computers	
	L–L	G–L	L–L	G–L	L–L	G–L	L–L	G–L
BL	76.83 ± 0.80	75.34 ± 0.43	93.10 ± 0.94	88.55 ± 0.43	94.81 ± 0.98	94.09 ± 0.83	**87.79 ± 0.94**	85.43 ± 0.23
BT	80.41 ± 0.15	—	**94.16 ± 0.02**	—	**96.55 ± 0.12**	—	86.86 ± 0.97	—
VICReg	**80.79 ± 0.12**	—	93.46 ± 0.08	—	95.59 ± 0.23	—	86.39 ± 0.32	—

(b) Graph classification performance

Obj.	PTC-MR			PROTEINS			REDDIT-B			IMDB-M		
	L–L	G–L	G–G	L–L	G–L	G–G	L–L	G–L	G–G	L–L	G–L	G–G
BL	60.32 ± 2.32	57.43 ± 2.31	61.05 ± 2.61	**75.32 ± 0.33**	72.86 ± 0.83	75.44 ± 1.48	76.32 ± 0.94	75.32 ± 0.84	77.70 ± 0.23	50.10 ± 0.32	49.23 ± 0.32	50.11 ± 0.12
BT	59.48 ± 2.09	—	57.58 ± 1.53	75.25 ± 1.11	—	75.08 ± 0.33	**84.97 ± 1.28**	—	**86.05 ± 0.69**	49.32 ± 0.54	—	49.82 ± 0.56
VICReg	**61.46 ± 0.89**	—	**61.89 ± 0.43**	75.10 ± 0.08	—	**75.47 ± 0.21**	83.70 ± 1.36	—	82.31 ± 0.23	**51.48 ± 0.15**	—	**51.46 ± 0.15**

We train the model with different contrasting modes, with topology augmentation set to ER and feature augmentation to FM, similar to our previous study. For other parameters, we closely follow common practices in their original papers. The experimental results are shown in Table 14. It should be noted that Barlow Twins (BT) and VICReg losses need to compute the covariance of latent embedding vectors, and thus these two objectives are not compatible with the global-local mode.

Obs. 7. Barlow Twins and VICReg losses reduce the computational burden without explicit negative samples and also obtain promising results.

Surprisingly, we find the performance obtained by employing these negative-sample-free objectives even surpasses their negative-sample-based counterparts, which suggests a promising future direction of more efficient solutions free of negative samples to GCL. As opposed to negative-sample-based objectives, the BL, BT, and VICReg losses eschew the need of explicit hard negative samples and thus greatly reduce the computational burden. To see this, we summarize the memory consumption in Table 15, from which we clearly observe that these three losses use much less memory than other objectives without negative samples.

Table 15: Memory usage (MB) on the PROTEINS dataset of different contrastive objectives.

Obj.	L–L	G–L	G–G
InfoNCE	6,311	2,977	2,271
JSD	6,309	2,845	2,269
TM	6,271	2,977	2,269
BL	2,235	2,247	2,187
BT	2,419	—	2,201
VICReg	2,465	—	2,232

C.3 Ablation Studies on Batch Normalization of the Bootstrapping Latent Loss

Previous work [62] empirically demonstrates that Batch Normalization (BN) compensates for improper initialization for contrastive learning networks, instead of introducing implicit negative samples. To further demonstrate this in GCL, we perform ablation studies by using BN or not in three critical components in the BL loss, i.e. the GNN encoder, the projector, and the predictor, on the node classification task. The results in Table 16 show that using BN in the GNN encoder *solely* is almost sufficient to obtain promising performance.

Table 16: Ablation studies on batch normalization of the bootstrapping latent loss. ✓ denotes having an extra BN layer in corresponding component.

Encoder	Projector	Predictor	Wiki	CS	Physics	Physics
✓	✓	✓	80.61 ± 00.04	93.29 ± 00.07	95.31 ± 00.01	89.81 ± 00.07
✓	✓	—	79.93 ± 00.06	93.08 ± 00.05	95.24 ± 00.01	88.64 ± 00.04
✓	—	✓	80.01 ± 00.03	92.93 ± 00.10	95.00 ± 00.04	87.42 ± 00.15
✓	—	—	79.54 ± 00.08	92.87 ± 00.04	95.11 ± 00.08	87.97 ± 00.20
—	✓	✓	79.42 ± 00.09	93.53 ± 00.03	95.23 ± 00.05	87.45 ± 00.11
—	✓	—	79.46 ± 00.03	92.82 ± 00.06	95.15 ± 00.02	88.32 ± 00.08
—	—	✓	79.96 ± 00.04	78.32 ± 00.69	85.91 ± 03.32	57.80 ± 00.33
—	—	—	78.73 ± 00.20	74.87 ± 00.18	58.50 ± 00.31	62.88 ± 00.08

D Background and Related Work

Recently, Self-Supervised Learning (SSL) has shown its great capability in alleviating the label scarcity problem. As a category of SSL, Contrastive Learning (CL) has attained increasing popularity due to its simplicity and powerful empirical performance. In essence, CL aims to learn discriminative representations by repulsing negative pairs and attracting positive pairs. Initially proposed for learning visual representations, visual CL work usually generates multiple views of the input images using data augmentation [63–66] at first. Under this multiview setting, congruent samples corresponding to one specific image in these views are considered as positive samples and other samples within the same batch [4, 8] or in an extra memory banks [5, 67] are used as negatives.

Graph Contrastive Learning (GCL) adapts the idea of CL to the graph domain. However, due to the complex, irregular structure of graph data, how to design strategies for constructing positive and negative samples for GCL is more challenging than visual data. Prior GCL work proposes data augmentation techniques for GCL, explores different contrasting modes to build contrastive pairs, and examines contrastive objectives to scoring positive and negative pairs. Regarding graph data augmentation, You et al. [16] study different data augmentation including node dropping, edge perturbation, subgraph sampling, and feature masking; MVGRL [14] employs graph diffusion to generate graph views with more global information; GCA [18] proposes adaptive augmentation techniques to further consider important topology and attribute information. Many other methods explore various contrasting modes using different parts of a graph. For example, GraphCL [16] and GRACE [17] contrasts node-node pairs, InfoGraph [19] considers graph-graph pairs, while DGI [13] and MVGRL [14] constructs node-graph contrasting pairs. Although there has been several survey papers on self-supervised graph representation learning [20–22], to the best of our knowledge, none of existing work provides rigorous empirical evidence on the impact of each component in GCL. In this work, we proposes a rather complete dissection of existing work and provides empirical insights into an effective GCL algorithm.

E Details of Design Dimensions

E.1 Data Augmentation

E.1.1 Topology Augmentation

Topology augmentation corrupts or perturbs the structural space of the graph by modifying its adjacency matrix $\boldsymbol{A}$.

Edge perturbation randomly adds and/or removes a portion of edges in the original graph. Formally, we sample a random masking matrix $\widetilde{\boldsymbol{R}} \in \{0, 1\}^{N \times N}$, where each entry is drawn from a Bernoulli distribution $\widetilde{\boldsymbol{R}}_{ij} \sim \mathrm{Bern}(p_r)$. Here p_r is the probability for adding/removing an edge. The resulting adjacency matrix can be computed as $\widetilde{\boldsymbol{A}} = \boldsymbol{A} \odot \widetilde{\boldsymbol{R}}$, where $\odot$ is a bit-wise operator. In this work, we consider three variants of edge perturbation, including **Edge Removing (ER)**, **Edge Adding (EA)**, and **Edge Flipping (EF)**, corresponding to $\odot$ being instantiated as bit-wise add, subtract, and exclusive or, respectively.

Node perturbation considers topology transformation in a node-wise manner. Because of the transductive nature of most GNN models, we mainly consider **Node Dropping (ND)** in this category. Similarly to ER, we assign each node with a probability of p_d being dropped. ND is equivalent to masking all adjacent edges to the dropped node to zeros in the adjacency matrix.

Subgraph sampling modifies the graph structure at the subgraph level. The sampling operation could either be lossless or lossy. For lossless sampling, we perform **Ego-net Sampling (ES)**, where each node and its k-hop neighbors are extracted to form a subgraph, given a k-layer graph encoder.[1] Besides ES, in this work, we are also concerned with **Subgraphs induced by Random Walks (RWS)**. Starting from a node, we sample a random walk that has a probability p_{ij} to travel from node v_i to v_j and a probability p_e to return to the start node [68]. Then, nodes appearing in this walk sequence are selected to construct a subgraph.

---

[1] Since a k-layer graph encoder is only able to extract information within k-hop neighborhoods, the ES process is essentially equivalent to an *identical mapping* to the graph structure.

Diffusion enriches the structure with more global information by adding new edges and changing edge weights. The resulting adjacency matrix takes a general form as

$$\widetilde{\boldsymbol{A}} = \sum_{k=0}^{\infty} \Theta_k \boldsymbol{T}^k, \tag{1}$$

where Θ is the weighting coefficient controlling the cooperation of global information with $\sum_{k=0}^{\infty} \Theta_k = 1$ and $\boldsymbol{T}$ is a generalized transition matrix computed from $\boldsymbol{A}$. We point out that the diffusion operator usually converts the original graph into a dense one, bringing heavy computation to graph convolutions. Therefore, in this paper, to ensure sparsity, we consider two sparse diffusion transformations: **Personalized PageRank (PPR)** [69] followed by hard thresholding as sparsification [70] and **Markov Diffusion Kernels (MDK)** [71, 72].

E.1.2 Feature Augmentation

Feature augmentation modifies to the attribute matrix $\boldsymbol{X}$. In this work, we concern the following two types of feature transformation functions.

Feature Masking (MF) randomly masks a fraction of dimensions with zeros in node features: $\widetilde{\boldsymbol{X}} = [\boldsymbol{x}_1 \circ \widetilde{\boldsymbol{m}};\ \boldsymbol{x}_2 \circ \widetilde{\boldsymbol{m}};\ \cdots;\ \boldsymbol{x}_N \circ \widetilde{\boldsymbol{m}}]^\top$, where $\circ$ is Hadamard product and $\boldsymbol{m} \in \{0,1\}^F$ is a random vector with each entry drawn from a Bernoulli distribution with a probability $(1 - p_m)$.

Instead masking node entries in a column-wise manner, we can also apply element-wise dropout [73] to the node feature matrix. In this **Feature Dropout (FD)** scheme, each entry has a probability p_f of being randomly masked with zero.

E.2 Contrasting Modes

For an anchor instance, Contrasting modes determine the positive and negative sets at different granularities of the graph. In mainstream work, three contrasting modes are widely employed.

- **Local-local CL** targets at contrasting between node-level representations in the two views. For a node embedding $\boldsymbol{v}_i$ being the anchor, the positive sample is its congruent counterpart in the other view $\boldsymbol{u}_i$; embeddings other than $\boldsymbol{u}_i$ are then naturally selected as negatives.
- **Global-local CL** enforces the compatibility between node- and graph-level embeddings. Specifically, for every global embedding $\boldsymbol{s}$ being the anchor instance, its the positive sample is all its node embedding $\boldsymbol{v}_i$. The global-local scheme shall be considered as a proxy for local-local CL, provided that the readout function r is expressive enough [36, 13]. Note that when only one graph is provided, we need an explicit corruption function (e.g., random shuffling) to construct negative samples from original node embeddings [13, 14].
- **Global-global CL** further achieves consistency between the graph embeddings of the two augmented views from the same graph. For a graph embedding $\boldsymbol{s}_1$, the positive sample is the embedding $\boldsymbol{s}_2$ of the other augmented view. In this case, other graph embeddings in the batch are considered as negative samples. This scheme can be applied to datasets with multiple graphs.

E.3 Contrastive Objectives

Contrastive objectives are used to train the encoder to maximize the agreement between positive samples and the discrepancy between negatives. We consider the following objective functions in this work.

- **InfoNCE** [74] gives a lower bound of MI up to a constant, defined as

  $$\mathcal{J}_{\text{InfoNCE}}(\boldsymbol{v}_i) = -\log \frac{e^{\theta(\boldsymbol{v}_i, P(\boldsymbol{v}_i))/\tau}}{e^{\theta(\boldsymbol{v}_i, P(\boldsymbol{v}_i))/\tau} + \sum_{\boldsymbol{q}_j \in \mathcal{Q}(\boldsymbol{v}_i)} e^{\theta(\boldsymbol{v}_i, \boldsymbol{q}_j)/\tau}}. \tag{2}$$

  Here $\theta(\cdot, \cdot)$ is a critic function measuring the similarity between two embeddings. Most work implements it with an additional projection head on embeddings followed by simple cosine similarity, i.e.

  $$\theta(\boldsymbol{u}, \boldsymbol{v}) = \frac{g(\boldsymbol{u})^\top g(\boldsymbol{v})}{\|g(\boldsymbol{u})\| \|g(\boldsymbol{v})\|},$$

where g is a multilayer perceptron.

- **Jason-Shannon Divergence (JSD)** computes the JS-divergence between the joint distribution of the product of marginals:

$$\mathcal{J}_{\text{JSD}}(\boldsymbol{v}_i) = \log d(\boldsymbol{v}_i, P(\boldsymbol{v}_j)) + \frac{1}{Q} \sum_{\boldsymbol{q}_j \in \mathcal{Q}(\boldsymbol{v}_i)} \log(1 - d(\boldsymbol{v}_i, \boldsymbol{q}_j)), \tag{3}$$

  where d is a discriminator function, which usually computes the inner product with a sigmoid activation:

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sigma(g(\boldsymbol{u})^\top g(\boldsymbol{v})).$$

  We kindly note that Hjelm et al. [3] employ a softplus version of the JS divergence:

$$\mathcal{J}_{\text{SP-JSD}}(\boldsymbol{v}_i) = -\operatorname{sp}(-d(\boldsymbol{v}_i, P(\boldsymbol{v}_j))) - \frac{1}{Q} \sum_{\boldsymbol{q}_j \in \mathcal{Q}(\boldsymbol{v}_i)} \operatorname{sp}(d(\boldsymbol{v}_i, \boldsymbol{q}_j)), \tag{4}$$

  where $\operatorname{sp}(x) = \log(1 + e^x)$.

- **Triplet Margin loss (TM)** directly enforces the *relative* distance between positive and negative pairs:

$$\mathcal{J}_{\text{TM}}(\boldsymbol{v}_i) = \max\left\{ \|\boldsymbol{v}_i - P(\boldsymbol{v}_i)\| - \frac{1}{Q} \sum_{\boldsymbol{q}_j \in \mathcal{Q}(\boldsymbol{v}_i)} \|\boldsymbol{v}_i - \boldsymbol{q}_j\| + \epsilon, 0 \right\}, \tag{5}$$

  where ϵ is the margin. TM is widely studied in metric learning literature [43].

Conceptually, these objective functions are related to the InfoMax principle [75], which aims to maximize the Mutual Information (MI) between representations of the same node in the two views. To be specific, InfoNCE and JSD are proved to be lower bounds of MI [76–78]; TM is also known to increase MI between positive representations but their relationship between MI maximization is not theoretically guaranteed. We further note that the InfoMax interpretation of these objectives may not be consistent with its actual behavior [79] and many recent studies provide theoretical understanding behind their success [80, 81].

In addition, we explore the following three contrastive objectives that rely on no explicit construction of negative samples:

- **Bootstrapping Latent loss (BL)** simply minimizes cosine similarity consistency between positive node embeddings without explicit negative samples [62, 7, 25, 82]:

$$\mathcal{J}_{\text{BL}}(\boldsymbol{v}_i) = -\frac{q(\boldsymbol{v}_i)^\top \boldsymbol{v}_i'}{\|q(\boldsymbol{v}_i)\| \|\boldsymbol{v}_i'\|}. \tag{6}$$

  Here $\boldsymbol{v}_i$ is the embedding from an online encoder on v_i while $\boldsymbol{v}_i'$ is the embedding obtained from an offline encoder on $P(v_i)$, and $q(\cdot)$ is a predictor attempting to predict $\boldsymbol{v}_i'$ from $\boldsymbol{v}_i$. We follow previous work [25] that symmetrize the architecture by also employing the online encoder on $P(v_i)$ and predicts an offline representation on v_i.

  Note that it is easy to see that directly optimizing Eq. (6) will result in a trivial solution that $q(\boldsymbol{v}_i) = \boldsymbol{v}_i'$. To avoid such model collapse, additional requirements are imposed such as asymmetric dual encoders, updating the offline encoder with exponential moving average [7], and batch normalization [83].

- **Barlow Twins loss (BT)** proposes to encourages similar representations between augmented views of a sample, while minimizing the redundancy *within* the latent representation vector [84, 85, 26]:

$$\mathcal{J}_{\text{BT}} = \sum_i (1 - \boldsymbol{C}_{ii})^2 + \lambda \sum_i \sum_{j \neq i} \boldsymbol{C}_{ij}^2, \tag{7}$$

  where $\boldsymbol{C}$ is the correlation matrix cross representations resulting from two augmented views and λ is a trade-off hyperparameter controlling the on- and off-diagonal terms.

- **VICReg loss** further combines variance and covariance regularization terms to Barlow Twins loss [86]:

$$\mathcal{J}_{\text{VICReg}} = \lambda s(\boldsymbol{V}, \boldsymbol{U}) + \mu(v(\boldsymbol{U}) + v(\boldsymbol{V})) + \gamma(c(\boldsymbol{U}) + c(\boldsymbol{V})), \tag{8}$$

  where $s(\cdot, \cdot)$ measures mean-squared Euclidean distance between each pair of vectors, $v(\cdot)$ is a hinge loss on the standard deviation of projections along the batch dimension, and $c(\cdot)$ defines a covariance term similar to BT. λ, μ, and γ control the importance of each term. Compared to the BT loss, VICReg is shown to be insensitive to any normalization tricks and much stabler than BT.

The latter BT and VICReg losses theoretically relate to the information bottleneck principle [85, 26], which learns representations being invariant to data augmentations while retaining informative about the sample itself [87, 88].

E.4 Negative Mining Strategies

Notwithstanding subtle differences in prior arts, existing work presumes embeddings of nodes or graphs other than the anchor instance to be dissimilar to the anchor and thus considers them as negatives. Hence it is natural to see that large batch/sampling sizes are needed for effective CL [5, 8] so as to include more negatives to provide more informative training signals.

In order to enrich the learning process with more information, many visual CL methods [89–94] advocate the explicit use of negative mining in the embedding space. Some methods develop debasing terms to select truly negative samples so as to avoid contrasting same-label instances; some other propose to upweight hard negative samples (points that are difficult to distinguish from an anchor) and remove easy ones that are less informative to improve the discriminative power of the GCL model.

In our benchmarking study, we consider the following four negative mining techniques:

- **Debiased Contrastive Learning (DCL)** [91] develops a debiased InfoNCE objective to correct for the sampling of same-label data points by reweighting positive and negative terms, motivated by the observation that sampling negative examples from truly different labels improves performance.
- **Hardness-Biased Negative Mining (HBNM)** [93] improves DCL by further introducing a hardness-biased objective with a user-predefined hardness level β. DCL could be regarded as a special case of HBNM when $\beta = 0$.
- **Hard Negative Mixing (HNM)** proposes to upweight hard negative samples by mixing up other hard samples [92, 95, 96]. Here we first select $2S$ hardest samples of the top-K similar to the anchor sample, denoted as $\{\boldsymbol{r}_i\}_{i=1}^{2S}$. Thereafter, we synthesize S samples for the anchor $\boldsymbol{v}_i$, where its hard negative samples are computed by

  $$\widetilde{\boldsymbol{v}}_i = \alpha_i \boldsymbol{v}_i + (1 - \alpha_i)\boldsymbol{r}_{i+s}, \tag{9}$$

  where α_i is sampled from a Beta distribution $\alpha_i \sim \mathrm{Beta}(1, 1)$.
- **Conditional Negative Sampling (CNS)** [94] proposes a ring-like negative sampling strategy to choose semi-hard negatives (that are not so hard and not so easy to an anchor sample) to yield strong representations. Specifically, CNS defines a lower and upper percentile l and u of pairwise distances to construct a support negative example set S_B.