Efficient Contrastive Learning for Fast and Accurate Inference on Graphs

Teng Xiao [1] **Huaisheng Zhu** [1] **Zhiwei Zhang** [1] **Zhimeng Guo** [1]
Charu C. Aggarwal [2] **Suhang Wang** [1] **Vasant G Honavar** [1]

Abstract

Graph contrastive learning has made remarkable advances in settings where there is a scarcity of task-specific labels. Despite these advances, the significant computational overhead for representation inference incurred by existing methods that rely on intensive message passing makes them unsuitable for latency-constrained applications. In this paper, we present `GraphECL`, a simple and efficient contrastive learning method for fast inference on graphs. `GraphECL` does away with the need for expensive message passing during inference. Specifically, it introduces a novel coupling of the `MLP` and `GNN` models, where the former learns to computationally efficiently mimic the computations performed by the latter. We provide a theoretical analysis showing why `MLP` can capture essential structural information in neighbors well enough to match the performance of `GNN` in downstream tasks. The extensive experiments on widely used real-world benchmarks that show that `GraphECL` achieves superior performance and inference efficiency compared to state-of-the-art graph constrastive learning (GCL) methods on homophilous and heterophilous graphs. Code is available at: `https://github.com/tengxiao1/GraphECL`.

1. Introduction

Over the past decade, there has been considerable interest in graph learning problems such as node classification, link prediction, and graph classification (Grover & Leskovec, 2016; Cui et al., 2018; Xu et al., 2019; Wu et al., 2020; Xiao et al., 2021; Chen et al., 2022; Xiao et al., 2024). Graph contrastive learning (GCL) has recently emerged as an attractive approach to graph representation learning in settings where there is a scarcity of task-specific labels (Zhu et al., 2021; Thakoor et al., 2021). On many key benchmarks, GCL has been shown to achieve performance that is competitive with or superior to that of state-of-the-art methods trained using ground truth labels (Zhu et al., 2021; Thakoor et al., 2021).

However, the significant computational overhead incurred by existing GCL methods, which rely on message passing for representation inference, limits their usefulness in latency-constrained applications. In particular, we observe that the state-of-the-art GCL methods achieve their superior performance using a graph neural network (GNN) encoder (see Figure 1). The message passing in GNN involves fetching the topology and features of numerous neighboring nodes to perform inference on a target node, which is computation-intensive during inference. Hence, there is an urgent need for inference-efficient alternatives to the state-of-the-art GCL methods. Recent work has begun to tackle the inference latency of `GNN` (Zheng et al., 2021; Zhang et al., 2021b; Tian et al., 2022; Wu et al., 2023), e.g., using knowledge distillation (KD) (Hinton et al., 2015) to learn an inference-efficient *student* `MLP` to mimick the output of a *teacher* `GNN`. However, they require task-specific labels to first train a good teacher `GNN`, limiting their applicability to GCL in settings where there is a lack of task-specific labels.

To the best of our knowledge, the following critical question, with important implications for real-world latency-constrained applications of GCL, remains unanswered: *How can we design a new GCL algorithm that outperforms state-of-the-art GCL methods on downstream tasks while avoiding high inference latency?* To answer this question, we present `GraphECL`, a *simple, effective, and efficient* contrastive regime on graphs. Specifically, to capture the graph structure of the nodes and achieve fast inference, `GraphECL` introduces a cross-model contrastive architecture in which positive examples consist of cross-model pairs (e.g., `MLP`-`GNN`) directly derived from neighborhood relations extracted from the graph. These positive samples are obtained from the representations of `MLP` and `GNN` of central nodes and their neighbors, respectively. This simple architecture allows `GraphECL` to benefit from the graph structure during training via `GNN` while using `MLP` to avoid relying on the graph structure during inference. Based on this cross-model architecture, we introduce a novel *general-*

[1]Penn State University, USA [2]IBM T. J. Watson Research Center, USA. Correspondence to: Teng Xiao <tengxiao@psu.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Figure 1. Inference latency v.s Accuracy on Pubmed. For GCL baselines, we test them on using both MLP and GNN backbones. We can observe that SOTA methods require GNN as the encoder to achieve good performance, which is computation-intensive during inference. GraphECL is fast with even higher accuracy

ized contrastive loss, which facilitates the learning of a computationally efficient MLP encoder, allowing the resulting model to effectively capture graph structural information so as to match the performance of state-of-the-art GCL methods on benchmark datasets, but without their prohibitive computational cost during inference.

Key Contributions. (i) We identify the key limitations of current GCL methods that limit their applicability to representation learning in latency-constrained real-world applications that require fast inference. (ii) We design GraphECL, a novel coupling of MLP and GNN models where the former learns to computationally efficiently mimic the computations performed by the latter. We show that the resulting model can effectively learn graph structural information and conduct fast inference with a simple MLP. (iii) We present theoretical analyses that offer insight into how GraphECL gradually encodes useful structural information using an MLP. Specifically, we show how GraphECL can theoretically achieve good generalization performance on downstream tasks. (iv) We demonstrate through extensive experiments that GraphECL can achieve ultra-fast inference speed and superior performance on downstream tasks simultaneously. Specifically, we show that the proposed GraphECL can run significantly faster than baselines on graphs, making it especially useful for latency-constrained applications where fast inference is a key requirement.

2. Related Work

Graph Contrastive Learning. GCL has emerged as a class of effective methods for learning useful representations from unlabeled graph data (Veličković et al., 2018; Zhu et al., 2021; Guo et al., 2024; Suresh et al., 2021; Zhang et al., 2021a; Zhu et al., 2024; Meng et al., 2019). Some authors have proposed freeing GCL from the need for negative samples (Thakoor et al., 2021; Zhang et al., 2021a) or even the need for graph augmentation (Xiao et al., 2022; Xiao & Wang, 2021; Lee et al., 2022; Zhang et al., 2022). Others have explored approaches to accelerate GCL training (Zheng et al., 2022b; Yang et al., 2022a; Han et al., 2022). Despite this progress, current methods of GCL incur significant computational overhead during inference, which limits their usefulness in latency-constrained real-world applications. This is largely due to their need to fetch neighbors and their associated features for a target node while performing inference (Zhang et al., 2021b). In this paper, we aim to address this limitation by avoiding the need for expensive message passing during inference by coupling MLP and GNN models so that the former MLP learns to computationally efficiently mimic the computations performed by GNN.

Learning MLPs on Graphs. Our work is also related to graph-regularized MLP (Yang et al., 2016; Hu et al., 2021; Yang et al., 2021b; Liu et al., 2020), which incorporates graph structure into MLPs through various auxiliary regularization terms inspired by traditional network embedding methods (Hamilton et al., 2017b; Grover & Leskovec, 2016; Tang et al., 2015). By implicitly encoding structural information into MLPs, one can enhance the representational power of MLP encoders while maintaining fast inference. It is worth noting that these methods, despite their differences, share a reliance on the strong homophily assumption (McPherson et al., 2001), which posits that one-hop neighbors of nodes that are linked should exhibit similar latent representations. Consequently, graph-regularized MLP significantly falls short of the performance achievable by GCL methods (Veličković et al., 2018; You et al., 2020), as shown in our experiments. GraphECL on the other hand, is designed to match the performance of state-of-the-art GCL methods, while achieving significantly faster inference.

Knowledge Distillation on Graphs. Knowledge distillation on graphs, which aims to distill pre-trained teacher GNNs into smaller student MLPs, has recently garnered significant attention (Yang et al., 2020; 2022b; Yan et al., 2020; Yang et al., 2021a; Joshi et al., 2022). Since student GNNs still require time-consuming message passing in inference, recent studies (Yang et al., 2023; Zhang et al., 2021b; Zheng et al., 2021; Tian et al., 2022; Wu et al., 2023) have shifted their focus towards GNN-MLP distillation. This involves learning an inference-efficient student MLP by distilling knowledge from the teacher GNN. However, these methods typically rely on task-specific labels to train the teacher GNN, which can be challenging in real-world scenarios where labels are often inaccessible. In contrast, our work aims to develop an inference-efficient and structure-aware MLP for faster inference in settings where task-specific labels are unavailable.

3. Preliminaries

Notations and Problem Setup. The input graph is denoted as $G = (V, E)$, where $V = \{v_1, \dots, v_{|V|}\}$ is a set of $|V|$

Figure 2. Existing contrastive schemes and GraphECL. (a) and (c) rely on invariant assumptions, aiming to learn augment-invariant representations of the same node. (b) is based on homophily assumptions, forcing neighboring nodes to exhibit same representations. In contrast, (d) showcases our GraphECL, which achieves significant inference efficiency and strong performance using learned MLP.

nodes and $\mathcal{E}$ denotes the set of edges. Each edge $e_{i,j} \in \mathcal{E}$ denotes a link between v_i and v_j. We use $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times D}$ to denote the node attributes, where i-th row of $\mathbf{X}$, i.e., $\mathbf{x}_i$, is the attribute vector of node v_i. The graph structure can be characterized by its adjacency matrix $\mathbf{A} \in [0,1]^{|\mathcal{V}| \times |\mathcal{V}|}$, where $\mathbf{A}_{i,j} = 1$ if there exists an edge $e_{i,j} \in \mathcal{E}$, and $\mathbf{A}_{i,j} = 0$ otherwise. Then, the graph $\mathcal{G}$ can be also denoted as a tuple of matrices: $\mathcal{G} = (\mathbf{X}, \mathbf{A})$. Given $\mathcal{G}$, our goal is to learn an efficient MLP encoder denoted by f_M with only attributes $\mathbf{X}$ as input, so that the inferred representation for node v: $\mathbf{v} = f_M(\mathbf{X})[v] \in \mathbb{R}^K$ is useful for downstream tasks. For brevity, in what follows, we omit the input $\mathbf{X}$ and use $f_M(v)$ to denote v's representation from MLP.

Graph Contrastive Learning (GCL) with Augmentations. GCL aims to learn representations (Trivedi et al., 2022; Veličković et al., 2018; Zhu et al., 2021; You et al., 2020; Suresh et al., 2021) by contrasting augmented views as presented in Figure 2 (a). Thus, for a given node v, its representation in an augmented view is trained to be similar to the representation of the same node v from another augmented view, while being distinct from the representations of other nodes, which serve as negative samples. Given two views $\mathcal{G}_1$ and $\mathcal{G}_2$, a widely-used contrastive objective is:

$$\mathcal{L}_{GCL} = \frac{1}{|\mathcal{V}|}\sum_{v \in \mathcal{V}} \log \ell(v), \text{ where} \tag{1}$$

$$\ell(v) \triangleq \frac{\exp(f_G(v^1)^\top f_G(v^2)/\tau)}{\exp(f_G(v^1)^\top f_G(v^2)/\tau) + \sum_{v^- \in \mathcal{V}^-} \exp(f_G(v^1)^\top f_G(v^-)/\tau)}.$$

Here $f_G(v^1) = f_G(\mathcal{G}_1)[v]$ and $f_G(v^2) = f_G(\mathcal{G}_2)[v]$ are GNN representations of the same node v from two views, where f_G denote the GNN encoder. $\mathcal{V}^-$ is the set of negative samples from inter- or intra- augmented view (Zhu et al., 2020). τ is the temperature. Although GCL with augmentations has achieved remarkable success, we note that such methods predominantly rely on the GNN encoder to capture structural invariances in different augmented views of the graph. This reliance is further discussed in Section 4.1 and results in substantial computational overhead during inference compared to MLP, as shown in Figure 1.

Graph-regularized MLP. Graph-MLP (Yang et al., 2016; Hu et al., 2021; Yang et al., 2021b; Liu et al., 2020) proposes to avoid the need for GNN neighbor fetching by learning an inference-efficient MLP model with a neighbor contrastive loss inspired by traditional graph embedding methods (Hamilton et al., 2017b; Grover & Leskovec, 2016; Tang et al., 2015). All such methods essentially minimize the following contrastive loss over neighbors in the graph:

$$\mathcal{L}_{GR} = \frac{1}{|\mathcal{V}|}\sum_{v \in \mathcal{V}} \frac{1}{|\mathcal{N}(v)|}\sum_{u \in \mathcal{N}(v)} \log \ell(v), \text{ where} \tag{2}$$

$$\ell(v) \triangleq \frac{\exp(f_M(v)^\top f_M(u)/\tau)}{\exp(f_M(v)^\top f_M(u)/\tau) + \sum_{v^- \in \mathcal{V}^-} \exp(f_M(v)^\top f_M(v^-)/\tau)};$$

where $f_M(v)$, $f_M(u)$ and $f_M(v^-)$ are projected representations by MLP of nodes v, u and v^-, respectively. $\mathcal{N}(v)$ denotes the set of positive examples containing local neighborhoods of the node v and $\mathcal{V}^-$ denotes the set of negative examples that are randomly sampled from $\mathcal{V}$. This approach is illustrated in Figure 2 (b). Despite its improved inference efficiency due to its exclusive use of MLP, this approach takes a significant hit in performance compared to GCL with augmentations (See Figure 1). Moreover, this scheme over-emphasizes homophily, assuming that nodes that are linked in the graph should have similar representations in the latent space, at the expense of structural information (You et al., 2020), making it difficult to generalize to graphs with heterophily (Lim et al., 2021). Table 6 in Appendix A details comparisons between current graph contrastive schemes and our GraphECL in terms of design assumptions, effectiveness (or representational power) and inference efficiency.

4. Efficient Graph Contrastive Learning

We proceed to introduce GraphECL, which aims to dramatically speed up inference while matching the performance of GCL. GraphECL adopts a cross-model contrastive architecture, wherein we design an asymmetric GNN-MLP architecture for the nodes and their neighbors to extract effective positive and negative examples in contrastive training (Section 4.1). To capture structural information in the graph, we introduce a generalized contrastive loss that extends the classic InfoNCE loss (Chen et al., 2020) from independent instance discrimination over augmentations to non-independent neighborhood contrast over graph structures, taking into account meaningful distance between neighboring nodes. Finally, we provide a theoretical analysis to show

Figure 3. Current GCL methods that employ a GNN-MLP architecture (where MLP is used for inference) exhibit a significant performance decay compared to those using a GNN-GNN architecture (where GNN is used for inference). We illustrate Pubmed as an example, though we observe the similar trend in other datasets.

that GraphECL can encode structural information to ensure good performance on downstream tasks (Section 4.2).

4.1. Simple Cross-model Contrastive Learning

As shown in Figure 1, the state-of-the-art GCL methods incur substantial computational overhead and hence inference latency due to the layer-wise message passing in GNN encoders. A straightforward idea is to replace the GNN encoders with the MLP encoders to speed up inference. However, as seen in Figure 1, the performance of current GCL methods using MLP instead of GNN is significantly worse than that of those using GNN. These results are consistent with what our expectations in that, while replacing GNN with MLP eliminates message passing and hence speeds up inference, does so by ignoring critical structural information from the graph. Thus, the encoder GNN plays a critical role in the success of GCL based on graph augmentations.

Cross-model Contrastive Architecture. To address this limitation, we first introduce a simple cross-model architecture of GraphECL. As Figure 1 shows, using MLP as the encoder for GCL achieves substantial speedup in inference but does so at the cost of substantial drop in performance. Conversely, utilizing GNN as the encoder yields superior performance but at the cost of significant slowdown during inference. Our solution to resolving this dilemma is elegantly simple, yet, as we will demonstrate, remarkably effective. Specifically, we employ a cross-model architecture with two encoders, one of which is a GNN, and the other an MLP. GNN in this architecture is exclusively dedicated to extracting and encoding structural information from the graph during the learning phase, whereas the MLP is used during the inference process to circumvent the need for computationally expensive message passing.

Can we directly apply this architecture to speed up inference in GCL? For instance, can we minimize the following cross-model InfoNCE-style loss, referred to as MLP-Augmented GCL loss for the architecture shown in Figure 2 (c)?

$$\mathcal{L}_{MA} = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}} \log \ell(v); \text{ where} \tag{3}$$

$$\ell(v) \triangleq \frac{\exp(f_G(v^1)^\top f_M(v^2)/\tau)}{\exp(f_G(v^1)^\top f_M(v^2)/\tau) + \sum_{v'\in\mathcal{V}} \exp(f_G(v^1)^\top f_M(v')/\tau)};$$

Unfortunately, the answer to this question is negative. As shown in Figure 3, even with the cross-model architecture, current GCL methods take a significant performance loss compared to those that use GNN encoders in Equation (3). In what follows, we introduce an effective and efficient contrastive learning loss that addresses this problem.

Efficient Contrastive Learning Loss on Graphs. Before proceeding to introduce the proposed contrastive loss, we first motivate it. Existing state-of-the-art GCL methods adopt graph augmentations that emphasize similarities in the encoding of a same node in different "augmented views", using GNN as the encoder (See Figure 2). In contrast, we want GraphECL to avoid relying on graph augmentations, but instead, learn an MLP representation of node by extracting and encoding its neighborhood structure and features from GNN. In particular, positive pairs in GCL are generated by random graph augmentations of the same node. In contrast, positive examples in GraphECL are cross-model pairs (e.g., MLP-GNN) directly provided by neighborhood relations present in graphs. These positive examples are obtained through MLP and GNN representations, respectively, of nodes and their neighbors. The preceding suggests a contrastive loss for GraphECL, minimizing which has the effect of pushing the MLP representation of each node closer to the GNN representations of its neighbors:

$$\mathcal{L}_{ECL} = \frac{1}{|\mathcal{V}|}\sum_{v\in\mathcal{V}} \frac{1}{|\mathcal{N}(v)|}\sum_{u\in\mathcal{N}(v)} \log \ell(v); \text{ where} \tag{4}$$

$$\ell(v) \triangleq \frac{\exp(f_M(v)^\top f_G(u)/\tau)}{\sum_{v'\in\mathcal{V}} \exp(f_G(u)^\top f_G(v')/\tau) + \lambda \exp(f_M(v)^\top f_G(v')/\tau)};$$

where $f_M(v)$ and $f_G(u)$ are the L2-normalized representations obtained from the MLP and GNN encoders, respectively, of node v and its neighbor u. Here, $(f_G(u), f_G(v'))$ and $(f_M(v), f_G(v'))$ represent intra-modael and inter-model negative pairs, respectively. v' is independently sampled as a negative example and λ serves as a hyperparameter to control the balance between the two types of negative pairs. For large graphs, we randomly sample M negative pairs for each node as an efficient approximation.

We highlight the benefits of using this simple alignment loss. First, the MLP encoder f_M can effectively preserve the local neighborhood distribution captured by GNN encoder f_G without the need for graph augmentation. GraphECL encodes the latent distributions (representations from GNN) of neighborhoods into the representation of central node from

Figure 4.(a) The toy graph where the color denotes node's semantic class. (b) Uni-model contrastive learning in GR-MLP encourages one-hop neighbors to have similar representations (c) Multi-model contrastive objective in GraphECL is not based on the one-hop homophily assumption but automatically captures graph structures based on different graphs beyond homophily. Thus, GraphECL exhibits robustness and generalizability on both homophilic and heterophilic graphs (See section 5).

MLP, enabling MLP to implicitly encode the structural information captured by GNN. As latent neighbors' representations encode high-order information through multi-layer message passing of GNN, MLP effectively distills the high-order structural information from the GNN. Second, in contrast to the GR-MLP model, which performs uni-model contrastive learning shown in Figure 4 (b), GraphECL aims to push cross-model contrastive learning, pushing the representation of a node $f_M(v)$ and that of its neighbors $f_G(u)$ close to each other. In particular, GraphECL does not necessarily imply the learned MLP representations $(f_M(v); f_M(u))$ become identical (homophily assumption). In other words, the multi-model signals ensure that node pairs v and v^0 with the same same neighborhood context serve as positive pairs for contrastive learning as illustrated in Figure 4 (c).

Interpretation. L_{ECL} is a simple yet very effective generalization of the popular InfoNCE loss in Equation (3) from uni-model instance discrimination over augmentations to cross-model contrast over graph neighbors. During the learning process, cross-model positive pairs of neighbors $(f_M(v), f_G(u))$ are pulled together in the latent space, while inter-modal $(f_M(v), f_G(v^-))$ and intra-model negative pairs $(f_G(u), f_G(v^-))$ are pushed apart. We empirically demonstrate in Section 5 that GraphECL generalizes as well as state-of-the-art GCL methods during the inference, with the additional benefit of significantly faster inference.

4.2. Theoretical Analysis

In this section, we provide theoretical evidence to support the design of our simple GraphECL. All detailed proofs can be found in Appendix B. We denote the normalized adjacency matrix $D^{-1}A$, with D being the diagonal degree matrix. We define the two representation metrics M and G where the v-th row $(M)_v = f_M(v)$ and the u-th row $(G)_u = f_G(u)$ represent the corresponding encoded representations from MLP and GNN, respectively. Let $\hat{A} = \exp(MG^\top/\tau)$ is the estimated affinity matrix based on representation similarity. $\hat{D} = \deg(\hat{A})$ is the diagonal matrix, whose element $(\hat{A})_{i,i}$ is the sum of the i-th row of $\hat{A}$. Next, we reveal the stationary point of the learning dynamics of GraphECL, which implies the equilibrium as:

Theorem 4.1. The learning dynamics w.r.t the MLP encoder f_M with efficient contrastive loss ($\lambda = 1$) in Equation (4) saturates when the true normalized adjacency and the estimated normalized affinity matrices agree: $D^{-1}A = \hat{D}^{-1}\hat{A}$, implying that, for $\forall v, u \in V$, we have:

$$P_n(u \mid v) = P_f(u \mid v), \quad \frac{\exp(f_M(v)^\top f_G(u)/\tau)}{\sum_{v^0 \in V} \exp(f_M(v)^\top f_G(v^0)/\tau)}; \qquad (5)$$

where $P_n(u \mid v)$ is the 1-hop neighborhood distribution (i.e., the v-th row of the normalized adjacency matrix) and $P_f(u \mid v)$ is the estimated neighborhood distribution.

Theorem 4.1 implies that GraphECL essentially learns a probabilistic model based on cross-modal encoders to predict the conditional 1-hop neighborhood distribution. Specifically, our assumption is more general than the homophily assumption. Even in heterophilic graphs, two nodes of the same semantic class tend to share similar structural roles, i.e., the 1-hop neighborhood context as shown in (Ma et al., 2021; Xiao et al., 2023) and statistics in Appendix C.2.

We also establish formal guarantees for the generalization of GraphECL on downstream tasks for learned MLP and GNN encoders. Without loss of generality, we use the linear probing task as an example. In this task, we train a linear classifier to predict class labels $y \in Y$ based on the MLP representation f_M using $g_{f,W}(v) = \arg\max_{c \in [C]} (f_M(v)^\top W)_c$, where $W \in \mathbb{R}^{K \times C}$ represents the weight matrix.

Theorem 4.2. Let f_M be the global minimum of generalized contrastive loss ($\lambda = 1$) in Equation (4) and $y(v)$ denote the label of v. $\lambda_1 \ge \cdots \ge \lambda_N$ are the eigenvalues with descending order of the normalized adjacency matrix $D^{-1}A$. Then, the linear probing error of f_M is upper-bounded by:

$$E(f_M), \min_W \frac{1}{|V|} \sum_{v \in V} 1[g_{f,W}(v) \neq y(v)] \le \frac{1-h}{1-\lambda_{K+1}}; \qquad (6)$$

where $h = \frac{1}{|V|} \sum_{v \in V} 1/|N(v)| \sum_{u \in N(v)} 1[y(v) = y(u)]$ and K is the dimensionality of the representation.

This theorem establishes a significant relationship between the downstream error in learned representations and two crucial factors: the parameter h and the $(K+1)$-th largest eigenvalue. Remarkably, h coincides precisely with the node homophily ratio metric (Pei et al., 2019; Lim et al., 2021). This metric calculates the proportion of a node's neighbors that share the same class label and then averages these values across all nodes within the graph. Homophilous graphs ($h \to 1$), exhibit a tendency for nodes to connect with others of the same class, while heterophilic graphs ($h \to 0$), display a preference for connections across different classes. This theorem shows that graphs characterized by a low homophily value (i.e., heterophilic graphs) may require a larger

Table 1. Node classification results (%) under the transductive setting on benchmarking homophilic and heterophilic graphs.

Datasets	Cora	Citeseer	Pubmed	Photo	WikiCS	Flickr	Cornell	Wisconsin	Texas	Actor
Graph-MLP	76.70±0.18	70.30±0.27	78.70±0.33	89.59±0.45	71.75±0.15	41.33±0.25	42.65±2.21	57.96±1.11	60.22±1.76	25.66±0.77
VGAE	76.30±0.21	66.80±0.23	75.80±0.40	91.50±0.20	72.19±0.31	40.71±0.22	48.73±4.19	55.67±1.37	50.27±2.21	26.99±1.56
DGI	82.30±0.60	71.80±0.70	76.80±0.60	91.61±0.22	75.35±0.14	44.70±0.26	45.33±6.11	55.21±1.02	58.53±2.98	28.30±0.76
GCA	82.93±0.42	72.19±0.31	80.79±0.45	91.70±0.10	78.35±0.05	46.10±0.19	52.31±1.09	59.55±0.81	52.92±0.46	28.77±0.29
SUGRL	83.40±0.50	73.00±0.40	81.90±0.30	93.07±0.15	79.83±0.31	46.22±0.31	50.18±0.30	61.31±2.07	57.88±2.21	<u>30.31±0.82</u>
BGRL	82.70±0.60	71.10±0.80	79.60±0.50	92.90±0.30	<u>79.98±0.10</u>	45.33±0.19	50.33±2.29	51.23±1.17	52.77±1.98	28.80±0.54
CCA-SSG	84.00±0.40	73.10±0.30	81.00±0.40	93.14±0.14	79.31±0.21	<u>47.54±0.14</u>	52.17±1.04	58.46±0.96	59.89±0.78	27.82±0.60
GGD	83.90±0.40	73.00±0.60	81.30±0.30	92.50±0.60	78.72±0.61	46.33±0.20	51.46±0.33	58.93±0.65	60.17±0.52	28.27±0.23
SGCL	82.97±0.20	72.58±0.25	81.25±0.32	<u>93.46±0.30</u>	79.85±0.53	46.35±0.18	53.28±1.37	59.93±0.75	61.26±0.65	26.51±0.47
GraphACL	<u>84.20±0.31</u>	73.63±0.22	82.02±0.15	93.31±0.19	78.75±0.36	46.95±0.27	<u>59.33±1.48</u>	<u>69.22±0.40</u>	<u>71.08±0.34</u>	30.03±0.13
AF-GCL	83.16±0.13	71.96±0.42	79.16±0.73	92.49±0.31	79.01±0.51	46.95±0.33	52.29±1.21	60.12±0.39	59.81±1.33	28.94±0.69
AFGRL	81.30±0.20	68.70±0.30	80.60±0.40	93.22±0.28	77.62±0.49	46.81±0.20	55.37±3.56	63.21±1.55	60.35±1.05	<u>30.31±0.95</u>
GraphECL	84.25±0.05	<u>73.15±0.41</u>	82.21±0.05	94.22±0.11	80.17±0.15	48.49±0.15	69.19±6.86	79.41±2.19	75.95±5.33	35.80±0.89

Table 2. Node classification results on large-scale graphs.

Datasets	snap-patents	ogbn-arxiv	ogbn-papers100M
BGRL	24.33±0.13	71.64±0.24	58.75±0.31
CCA-SSG	25.51±0.46	71.21±0.20	57.31±0.18
GraphACL	<u>26.18±0.39</u>	<u>71.72±0.26</u>	59.35±0.27
SUGRL	25.11±0.32	69.30±0.20	<u>60.31±0.22</u>
SGCL	24.91±0.46	70.99±0.09	59.96±0.37
GraphECL	27.22±0.06	71.75±0.22	61.45±0.31

representation dimension, i.e., smaller $(K+1)$-th largest $_{K+1}$ to effectively bound the downstream error.

5. Experiments

Datasets. We use established benchmarks for homophilic graphs: Cora, Citeseer, Pubmed, Photo, WikiCS, and Flickr, and for heterophilic graphs: Cornell, Wisconsin, Texas, and Actor. Additionally, we evaluate GraphECL on large-scale graphs, specifically the heterophilic Snap-patents, and homophilic Ogbn-arxiv and Ogbn-papers100M. In all datasets, we use the standard splits used in prior studies (Zhang et al., 2021a). The dataset details, splits, and statistics are in C.3.

Baselines. We compare GraphECL with the following graph contrative learning methods: Graph-MLP (Hu et al., 2021), VGAE (Kipf & Welling, 2016), DGI (Veličković et al., 2018), GCA (Zhu et al., 2021), SUGRL (Mo et al., 2022), BGRL (Thakoor et al., 2021), CCA-SSG (Zhang et al., 2021a), AF-GCL (Wang et al., 2022), AFGRL (Lee et al., 2022), GGD (Zheng et al., 2022b), GraphACL (Xiao et al., 2023), and SGCL (Sun et al., 2024).

Evaluation Protocol. Following (Velickovič et al., 2018; Thakoor et al., 2021), we consider three downstream tasks: node classification and graph classification. We use standard linear-evaluation protocol, where a linear classifier is trained on top of the frozen node or graph representations, and test accuracy is used as a proxy for representation quality.

Transductive vs. Inductive. Evaluation of node representations obtained using unsupervised learning through transductive node classification is a prevalent practice in the GCL literature. However, such evaluation neglects the scenarios of inferring representations for previously unseen nodes. Thus, it can not evaluate the real-world applicability of a deployed model, which often requires the inference of representations of novel nodes. Hence, following (Zhang et al., 2021b), we consider evaluation of learned representations under two settings: transductive (tran) and inductive (ind). The details about these two settings are in Appendix C.4.

Setup. For a fair comparison, we employ a standard GCN model (Kipf & Welling, 2017) as the GNN encoder for full-batch training on small graphs. For large-scale graphs, we use Graphsage (Hamilton et al., 2017a) with the subgraph sampling strategy in a mini-batch manner. For ogbn paper datasets, we utilize all-roberta-large-v1 (Reimers & Gurevych, 2019; Duan et al., 2023) as the feature extractor. We conduct experiments using several random seeds and report both the average performance and standard deviation. We select the optimal hyperparameters solely based on accuracy on the validation set. In cases where publicly available and standardized data splits were used in the original paper, we adopt their reported results. For baselines that deviated from standardized data splits, we either reproduce the results using the authors' official code. The details of hyperparameter search are provided in Appendix C.5.

5.1. Main Results and Comparison on both Transductive and Inductive Settings

In this section, we evaluate the node representations from the MLP encoder learned by our GraphECL.

Transductive Setting. We first consider the standard transductive setting in the task of node classification. We provide the results of other tasks in Table 8 in Appendix, respectively. Table 1 reports the average accuracy on both heterophilic

Table 3. Node classification results in a real-world scenario with both inductive and transductive nodes. *tran* denotes the accuracy on seen test transductive nodes. *ind* indicates the accuracy on unseen test inductive nodes.

Methods	Citeseer		Pubmed		Photo		Actor		Flickr		ogbn-arxiv	
	tran	ind	tran	ind	tran	ind	tran	ind	tran	ind	tran	ind
DGI	$63.82_{\pm1.69}$	$66.25_{\pm2.54}$	$70.33_{\pm2.61}$	$70.48_{\pm2.41}$	$87.11_{\pm1.65}$	$88.14_{\pm0.45}$	$28.07_{\pm2.19}$	$28.08_{\pm1.96}$	$37.84_{\pm0.22}$	$39.71_{\pm0.30}$	$65.21_{\pm0.35}$	$63.91_{\pm0.37}$
GCA	$66.33_{\pm1.16}$	$69.02_{\pm2.08}$	$\underline{81.16}_{\pm0.80}$	$\underline{81.52}_{\pm0.56}$	$\underline{90.54}_{\pm0.54}$	$90.59_{\pm0.51}$	$27.94_{\pm1.62}$	$27.72_{\pm1.51}$	$41.25_{\pm0.33}$	$42.95_{\pm0.18}$	$67.15_{\pm0.29}$	$66.95_{\pm0.25}$
BGRL	$67.04_{\pm1.44}$	$67.62_{\pm1.24}$	$78.36_{\pm0.41}$	$79.55_{\pm0.40}$	$87.95_{\pm0.68}$	$88.30_{\pm0.45}$	$\underline{29.04}_{\pm1.06}$	$\underline{29.07}_{\pm0.65}$	$40.78_{\pm0.20}$	$41.75_{\pm0.15}$	$68.57_{\pm0.31}$	$67.11_{\pm0.29}$
SUGRL	$\underline{69.16}_{\pm0.63}$	$\underline{71.24}_{\pm1.06}$	$81.07_{\pm0.76}$	$80.52_{\pm1.21}$	$89.88_{\pm0.64}$	$89.11_{\pm0.24}$	$28.95_{\pm1.37}$	$28.68_{\pm1.18}$	$40.37_{\pm0.20}$	$41.33_{\pm0.25}$	$\underline{69.96}_{\pm0.39}$	$\underline{68.21}_{\pm0.37}$
CCA-SSG	$68.81_{\pm1.05}$	$70.05_{\pm2.70}$	$79.76_{\pm2.32}$	$80.34_{\pm2.32}$	$88.60_{\pm1.95}$	$88.77_{\pm1.85}$	$28.52_{\pm1.11}$	$28.06_{\pm2.69}$	$\underline{42.16}_{\pm0.25}$	$\underline{43.22}_{\pm0.27}$	$68.34_{\pm0.17}$	$67.72_{\pm0.29}$
GraphECL	$69.96_{\pm0.10}$	$72.87_{\pm1.30}$	$81.71_{\pm0.91}$	$82.47_{\pm1.00}$	$92.18_{\pm0.15}$	$\underline{89.42}_{\pm0.03}$	$36.18_{\pm1.29}$	$37.17_{\pm1.84}$	$45.43_{\pm0.14}$	$43.50_{\pm0.20}$	$70.58_{\pm0.23}$	$70.12_{\pm0.12}$

Table 4. Ablation studies on Flicker dataset.

Ablation	Accuracy (%)
A1 w/o inter-model negative pair	$42.34_{\pm0.03}$
A2 w/o intra-model negative pair	$42.34_{\pm0.01}$
A3 w/o both types of negative pairs	$40.25_{\pm0.05}$
A4 w/ only MLP encoder	$\underline{44.83}_{\pm0.06}$
A1 & A4	$42.34_{\pm0.04}$
A2 & A4	$42.32_{\pm0.10}$
A3 & A4	$41.28_{\pm0.02}$
GraphECL	$48.49_{\pm0.15}$

and homophilic graphs. As shown in the table, across different datasets, GraphECL can learn representations that outperform other methods. These results are indeed remarkable, given that GraphECL exclusively employs the learned MLP representations for inference without any reliance on input graph structures. This demonstrates that MLP learned by GraphECL is able to capture meaningful structural information that is beneficial and generalized to downstream tasks. Since the inference time is much more heavily weighted in the large-scale graphs, we compare GraphECL with baselines in large-scale graphs in Table 2. From the table, we can observe that our efficient GraphECL can still achieve better performance on large-scale graphs.

Inductive Setting. To gain a better understanding of GraphECL's effectiveness, we evaluate the representations in a realistic production scenario that encompasses both transductive and inductive settings. In inductive evaluation, we set aside certain test nodes (20%) from test nodes in the transductive setting to create an inductive set (see Section C.4). We adopt GraphSAGE (Hamilton et al., 2017a) as the encoder during training for all methods. As shown in Table 3, GraphECL still achieves superior or competitive performance compared to elaborate methods employing GNN as the inference encoder. These results support the deployment of MLP learned by GraphECL as a significantly faster model, with minimal or no performance degradation.

5.2. Inference Time Comparison on Large-Scale Graphs

To demonstrate the inference efficiency of GraphECL, we compare the inference time on large-scale graphs with BGRL (GNN-L2W256), CCA-SSG (GNN-L2W128), and SUGRL (GNN-L1W128), where GNN-LiWj indicates the method achieving the best performance with i layers of GraphSAGE with dimension j, as shown in Table 1. The

Figure 5. Inference time comparison of different methods on large-scale graphs. Note that time axes are log-scaled.

MLP learned by GraphECL has 128 hidden dimensions. Our results in Table 1 and Figure 5 indicate that GraphECL achieves the highest accuracy while attaining significant speedups in inference. In large-scale graphs, the MLP learned by GraphECL is about 200x faster than CCA-SSG with the same number of layers, which demonstrates the superior inference efficiency of GraphECL.

5.3. Ablation Studies and Further Model Analysis

Ablation Studies. We study the effects of intra-model and inter-model negative losses. We consider three ablations: (A1) Removing the inter-model negative pairs; (A2) Removing the intra-model negative pairs; and (A3) Removing both intra-model and inter-model negative pairs. We also explore the effects of the cross-model contrastive architecture in GraphECL by removing the asymmetric GNN-MLP architecture and consider other two ablations for GraphECL: (A4) using only the MLP as the encoder. Table 4 lists the results. We also find that the performance of GraphECL suffers in the absence of negative examples, which shows that the information provided by negative examples is crucial for good generalization. Additionally, we observe that GraphECL using only MLP as the encoder can not match the performance of SOTA methods, although it can achieve fast inference. These results collectively underscore the importance of each of the components of GraphECL's GNN-MLP architecture in achieving SOTA performance on downstream tasks while achieving substantial inference speedup.

Effectiveness of Generalized Contrastive Loss. We maintain the cross-model contrastive architecture while replacing our generalized contrastive loss with the vanilla InfoNCE loss, as shown in Equation (3). Table 5 summarizes the

Figure 6. (Left) The training dynamics and inference performance on Cora. (Right two) The pairwise cosine similarity of representations for randomly sampled node pairs, one-hop neighbors, and two-hop neighbors on Cora and Actor. More results are in Appendix D.5.

Table 5. Ablation study on the effectiveness of using cross-model contrastive loss in GraphECL.

	Cora	Citeseer	Pubmed	Photo	Actor
GraphECL (InfoNCE)	74.55±0.45	67.15±0.25	76.50±1.20	91.44±0.61	33.39±0.37
GraphECL (Generalized)	84.25±0.05	73.15±0.41	82.21±0.05	94.22±0.11	35.80±0.89

results across all six datasets, demonstrating consistent improvements when using the generalized contrastive loss (Equation (4)) in our GraphECL formulation.

Size of Negative Pairs We investigate the influence of different number of negative pairs (i.e., M) in GraphECL (Appendix D.3). While a proper range can lead to certain gains (Figure 9), a small number of negative samples (e.g., $M = 5$) is enough to achieve good performance.

Dimensionality of Representation. We study the effects of dimensionality representation (i.e., K) in Appendix D.4. Not surprisingly, we find that larger dimensions often yield better results, with performance leveling off or decreasing when dimensionality becomes very large, for both homophilic and heterophilic graphs. This observation is consistent with Theorem 4.2, showing that a larger dimension can effectively reduce the upper bound of downstream errors.

The Parameter λ. We explore the effects of the trade-off parameter of λ and message passing layers during training. As shown in Figure 8, while a specific value can lead to certain gains, GraphECL is robust to different choices of the value λ on different graphs and our GraphECL is not very sensitive to λ used in training.

Training Dynamics. We also investigate the training process of GraphECL. Figure 6 (left) shows the curves of training losses and downstream performance using GNN and MLP, respectively. We find that: (1) GraphECL exhibits training stability, consistently improving performance as training losses decrease; (2) As the training proceeds, MLP gradually and dynamically acquires knowledge from GNN, facilitating the dynamic exchange of information between cross-model GNN and MLP in GraphECL.

Visualization. In addition to quantitative analysis, we visualize pairwise cosine similarities among randomly sampled nodes, one-hop neighbors, and two-hop neighbor pairs based on learned representations. Figure 6 shows that, in the homophilic graph (i.e., Cora), nodes exhibit representations that are similar to those of their neighbors. GraphECL enhances similarities between neighbor nodes compared to randomly sampled node pairs, demonstrating its ability to effectively preserve one-hop neighborhood contexts. In addition, in the heterophilic graph (i.e., Actor), GraphECL strives to bring two-hop neighbor nodes closer together. This observation is consistent with our analytical insights, showing that GraphECL is effective at automatically capturing regularities in graph structures beyond just homophily.

6. Conclusion

In this paper, we introduced GraphECL, a simple, novel, effective and inference-efficient GCL framework for learning effective node representations from graph data. GraphECL introduces a cross-model contrastive architecture and a generalized contrastive loss to train MLP encoder. GraphECL is faster, often by orders of magnitude, than GCL methods using the GNN encoder, while also achieving superior performance! We demonstrate theoretically that GraphECL leverages neighborhood distribution as an inductive bias. Extensive experiments on real-world small and large-scale graphs demonstrate its advantages over current methods, including the vastly superior inference efficiency and generalization on both homophilic and heterophilic graphs.

While our approach relies on node attributes, assuming access to these attributes is a modest assumption. In real-world applications, node attributes are often high-dimensional and rich in information. Therefore, an MLP optimized with our GraphECL can effectively distill structural information from the GNN even without any labels. Addressing the challenge of dealing with graphs lacking node attributes is a promising future direction, potentially involving an early layer or other pre-processing network embedding methods.

Acknowledgements

This work was funded in part by grants from the National Science Foundation (2226025, 2041759), the National Center for Advancing Translational Sciences, and the National Institutes of Health (UL1 TR002014).

Impact Statement

This paper advances contrastive learning on graphs. By improving the inference-efficiency and accuracy of contrastive learning on graphs, this research has the potential to contribute to advancements in latency-constrained applications. This work shares the societal implications of machine learning broadly, without raising any additional concerns.

References

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607, 2020.

Chen, Z., Xiao, T., and Kuang, K. Ba-gnn: On learning bias-aware graph neural network. In *2022 IEEE 38th International Conference on Data Engineering*, pp. 3012–3024, 2022.

Chien, E., Chang, W.-C., Hsieh, C.-J., Yu, H.-F., Zhang, J., Milenkovic, O., and Dhillon, I. S. Node feature extraction by self-supervised multi-scale neighborhood prediction. In *International Conference on Learning Representations*, 2021.

Cui, P., Wang, X., Pei, J., and Zhu, W. A survey on network embedding. *IEEE transactions on knowledge and data engineering*, pp. 833–852, 2018.

Duan, K., Liu, Q., Chua, T.-S., Yan, S., Ooi, W. T., Xie, Q., and He, J. Simteg: A frustratingly simple approach improves textual graph learning. *arXiv preprint arXiv:2308.02565*, 2023.

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022.

Eckart, C. and Young, G. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 855–864, 2016.

Guo, X., Wang, Y., Wei, Z., and Wang, Y. Architecture matters: Uncovering implicit mechanisms in graph contrastive learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Guo, X., Wang, Y., Wei, Z., and Wang, Y. Architecture matters: Uncovering implicit mechanisms in graph contrastive learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 2017a.

Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017b.

Han, X., Zhao, T., Liu, Y., Hu, X., and Shah, N. Mlpinit: Embarrassingly simple gnn training acceleration with mlp initialization. In *The Eleventh International Conference on Learning Representations*, 2022.

HaoChen, J. Z., Wei, C., Gaidon, A., and Ma, T. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *arXiv preprint arXiv:2106.04156*, 2021.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, pp. 22118–22133, 2020.

Hu, Y., You, H., Wang, Z., Wang, Z., Zhou, E., and Gao, Y. Graph-mlp: Node classification without message passing in graph. *arXiv preprint arXiv:2106.04051*, 2021.

Joshi, C. K., Liu, F., Xun, X., Lin, J., and Foo, C. S. On representation knowledge distillation for graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Lee, N., Lee, J., and Park, C. Augmentation-free self-supervised learning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7372–7380, 2022.

Lim, D., Hohne, F., Li, X., Huang, S. L., Gupta, V., Bhalerao, O., and Lim, S. N. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. Advances in neural information processing systems, pp. 20887–20902, 2021.

Liu, Q., Zhang, H., and Liu, Z. Simplification of graph convolutional networks: A matrix factorization-based perspective. arXiv preprint arXiv:2007.09036, 2020.

Liu, Y., Zheng, Y., Zhang, D., Lee, V. C., and Pan, S. Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. In Proceedings of the AAAI conference on artificial intelligence, pp. 4516–4524, 2023.

Ma, Y., Liu, X., Shah, N., and Tang, J. Is homophily a necessity for graph neural networks? In International Conference on Learning Representations, 2021.

McAuley, J., Targett, C., Shi, Q., and Van Den Hengel, A. Image-based recommendations on styles and substitutes. In ACM SIGIR conference on research and development in information retrieval, pp. 43–52, 2015.

McPherson, M., Smith-Lovin, L., and Cook, J. M. Birds of a feather: Homophily in social networks. Annual review of sociology, 27(1):415–444, 2001.

Meng, E. and Liu, Y. Graph contrastive learning with graph info-min. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, pp. 4195–4199, 2023.

Meng, Z., Liang, S., Fang, J., and Xiao, T. Semi-supervisedly co-embedding attributed networks. Advances in neural information processing systems, 32, 2019.

Mernyei, P. and Cangea, C. Wiki-cs: A wikipedia-based benchmark for graph neural networks. arXiv preprint arXiv:2007.02901, 2020.

Mo, Y., Peng, L., Xu, J., Shi, X., and Zhu, X. Simple unsupervised graph representation learning. In AAAI Conference on Artificial Intelligence, 2022.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. Geom-gcn: Geometric graph convolutional networks. In International Conference on Learning Representations, 2019.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084, 2019.

Sun, W., Li, J., Chen, L., Wu, B., Bian, Y., and Zheng, Z. Rethinking and simplifying bootstrapped graph latents. ACM international conference on web search and data mining, 2024.

Suresh, S., Li, P., Hao, C., and Neville, J. Adversarial graph augmentation to improve graph contrastive learning. Advances in Neural Information Processing Systems, 34: 15920–15933, 2021.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-scale information network embedding. In The Web Conference, pp. 1067–1077, 2015.

Thakoor, S., Tallec, C., Azar, M. G., Azabou, M., Dyer, E. L., Munos, R., Veličković, P., and Valko, M. Large-scale representation learning on graphs via bootstrapping. In International Conference on Learning Representations, 2021.

Tian, Y., Zhang, C., Guo, Z., Zhang, X., and Chawla, N. Learning mlps on graphs: A unified view of effectiveness, robustness, and efficiency. In The Eleventh International Conference on Learning Representations, 2022.

Trivedi, P., Lubana, E. S., Heimann, M., Koutra, D., and Thiagarajan, J. Analyzing data-centric properties for graph contrastive learning. Advances in Neural Information Processing Systems, pp. 14030–14043, 2022.

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. In International Conference on Learning Representations, 2018.

Wang, H., Zhang, J., Zhu, Q., and Huang, W. Augmentation-free graph contrastive learning with performance guarantee. arXiv preprint arXiv:2204.04874, 2022.

Wu, L., Lin, H., Huang, Y., , and Li, S. Z. Quantifying the knowledge in gnns for reliable distillation into mlps. In International Conference on Machine Learning, PMLR, 2023.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems, pp. 4–24, 2020.

Xia, J., Wu, L., Wang, G., Chen, J., and Li, S. Z. Progcl: Rethinking hard negative mining in graph contrastive learning. In International Conference on Machine Learning, pp. 24332–24346, 2022.

Xiao, T. and Wang, D. A general offline reinforcement learning framework for interactive recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4512–4520, 2021.

Xiao, T., Chen, Z., Wang, D., and Wang, S. Learning how to propagate messages in graph neural networks. In

Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021.

Xiao, T., Chen, Z., Guo, Z., Zhuang, Z., and Wang, S. Decoupled self-supervised learning for graphs. Advances in Neural Information Processing Systems, pp. 620–634, 2022.

Xiao, T., Zhu, H., Chen, Z., and Wang, S. Simple and asymmetric graph contrastive learning without augmentations. Advances in neural information processing systems, 2023.

Xiao, T., Cui, C., Zhu, H., and Honavar, V. G. Molbind: Multimodal alignment of language, molecules, and proteins. arXiv preprint arXiv:2403.08167, 2024.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In International Conference on Learning Representations, 2019.

Yan, B., Wang, C., Guo, G., and Lou, Y. Tinygnn: Learning efficient graph neural networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1848–1856, 2020.

Yang, C., Liu, J., and Shi, C. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In Proceedings of the web conference 2021, pp. 1227–1237, 2021a.

Yang, C., Wu, Q., Wang, J., and Yan, J. Graph neural networks are inherently good generalizers: Insights by bridging gnns and mlps. In The Eleventh International Conference on Learning Representations, 2022a.

Yang, C., Wu, Q., and Yan, J. Geometric knowledge distillation: Topology compression for graph neural networks. Advances in Neural Information Processing Systems, 35: 29761–29775, 2022b.

Yang, H., Ma, K., and Cheng, J. Rethinking graph regularization for graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4573–4581, 2021b.

Yang, L., Tian, Y., Xu, M., Liu, Z., Hong, S., Qu, W., Zhang, W., Cui, B., Zhang, M., and Leskovec, J. Vqgraph: Graph vector-quantization for bridging gnns and mlps. arXiv preprint arXiv:2308.02117, 2023.

Yang, Y., Qiu, J., Song, M., Tao, D., and Wang, X. Distilling knowledge from graph convolutional networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7074–7083, 2020.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In International conference on machine learning, pp. 40–48. PMLR, 2016.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. Advances in neural information processing systems, pp. 5812–5823, 2020.

You, Y., Chen, T., Wang, Z., and Shen, Y. Bringing your own view: Graph contrastive learning without prefabricated data augmentations. In ACM international conference on web search and data mining, pp. 1300–1309, 2022.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. K. Graphsaint: Graph Sampling Based Inductive Learning Method. In International Conference on Learning Representations, 2020.

Zhang, H., Wu, Q., Yan, J., Wipf, D., and Yu, P. S. From canonical correlation analysis to self-supervised graph neural networks. Advances in Neural Information Processing Systems, 2021a.

Zhang, S., Liu, Y., Sun, Y., and Shah, N. Graph-less neural networks: Teaching old mlps new tricks via distillation. In International Conference on Learning Representations, 2021b.

Zhang, Y., Zhu, H., Song, Z., Koniusz, P., and King, I. Costa: covariance-preserving feature augmentation for graph contrastive learning. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2524–2534, 2022.

Zheng, W., Huang, E. W., Rao, N., Katariya, S., Wang, Z., and Subbian, K. Cold brew: Distilling graph node representations with incomplete or missing neighborhoods. In International Conference on Learning Representations, 2021.

Zheng, X., Liu, Y., Pan, S., Zhang, M., Jin, D., and Yu, P. S. Graph neural networks for graphs with heterophily: A survey. arXiv preprint arXiv:2202.07082, 2022a.

Zheng, Y., Pan, S., Lee, V., Zheng, Y., and Yu, P. S. Rethinking and scaling up graph contrastive learning: An extremely efficient approach with group discrimination. Advances in Neural Information Processing Systems, 35: 10809–10820, 2022b.

Zhu, H., Xiao, T., and Honavar, V. G. 3m-diffusion: Latent multi-modal diffusion for text-guided generation of molecular graphs. arXiv preprint arXiv:2403.07179, 2024.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131, 2020.

Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., and Wang, L. Graph contrastive learning with adaptive augmentation. In The Web Conference, pp. 2069–2080, 2021.

Table 6. Comparison to previous contrastive schemes on graphs. Our approach does not rely on the invariant assumption that the augmentation can preserve the semantic nature of samples or the homophily assumption that connected nodes have similar representations.

Contrastive Schemes	Invariant Assumption	Homophily Assumption	Task Effective	Inference Efficient
GCL with Augmentations	3	7	3	7
Graph-MLP	7	3	7	3
MLP-Augmented GCL	3	7	7	3
Our GraphECL	7	7	3	3

A. Additional Related Work

In this section, we provide a detailed comparison of characteristics with previous contrastive schemes on graphs in Table 6.

Comparisons with GCL methods with augmentation (Trivedi et al., 2022; Veličković et al., 2018; Zhu et al., 2021; You et al., 2020; 2022; Suresh et al., 2021; Xia et al., 2022; Meng & Liu, 2023; Guo et al., 2023): This contrastive learning scheme relies on graph augmentations and is built based on the invariant assumption that the augmentation can preserve the semantic nature of samples, i.e., the augmented samples have invariant semantic labels with the original ones. This scheme requires a GNN as the encoder to achieve good performance, which is computation-intensive during inference. However, our GraphECL is not based on graph augmentations but directly captures the 1-hop neighborhood distribution.

Comparisons with Graph-MLP (Yang et al., 2016; Hu et al., 2021; Yang et al., 2021b; Liu et al., 2020): Despite its inference efficiency due to the exclusive use of MLP, this approach exhibits significantly lower performance compared to GCL, as depicted in Figure 1. Moreover, this scheme over-emphasizes homophily (You et al., 2020; Xiao et al., 2022), making it difficult to generalize to graphs with heterophily (Liu et al., 2023; Xiao et al., 2022; Zheng et al., 2022a). In contrast, GraphECL enjoys good downstream performance with fast inference speed for both homophilic and heterophilic graphs.

Comparisons with MLP-augmented GCL: This method mentioned in Section 4.1 also relies on the invariant assumption that augmentation can preserve the semantic nature of samples, i.e., the augmented samples have invariant semantic labels with the original ones. In addition, it suffers from significant performance degradation on downstream tasks when using MLP encoder for inference as shown in the results in Section 4.1.

B. Proofs

B.1. Proofs of Theorem 4.1

Theorem 4.1. The learning dynamics w.r.t the MLP encoder f_M with the generalized contrastive loss ($\epsilon = 1$) in Equation (4) saturates when the true normalized adjacency and the estimated normalized affinity matrices agree: $\tilde{A} := D^{-1}A$, which implies that, for $\forall v, u \in V$, we have:

$$P_n(u \mid v) = P_f(u \mid v) \triangleq \frac{\exp(f_M(v)^\top f_G(u)/\tau)}{\sum_{v' \in V} \exp(f_M(v)^\top f_G(v')/\tau)}, \tag{7}$$

where $P_n(u \mid v)$ is the 1-hop neighborhood distribution (i.e., the v-th row of the normalized adjacency matrix) and $P_f(u \mid v)$ is the estimated neighborhood distribution.

Proof. We first show that minimizing GraphECL objective with $\epsilon = 1$ is approximately to minimizing the losses of the positive and negative pairs on MLP representations.

$$\begin{aligned}
\mathcal{L}_{ECL} &= -\frac{1}{|V|}\sum_{v \in V}\frac{1}{|N(v)|}\sum_{u \in N(v)} \log \frac{\exp(f_M(v)^\top f_G(u)/\tau)}{\sum_{v^- \in V} \exp(f_G(u)^\top f_G(v^-)/\tau) + \exp(f_M(v)^\top f_G(v^-)/\tau)}; \\
&\approx \frac{1}{|V|}\sum_{v \in V}\Big(-\frac{1}{|N(v)|}\sum_{u \in N(v)} f_M(v)^\top f_G(u)/\tau + \log \sum_{v^- \in V} \exp(f_M(v)^\top f_G(v^-)/\tau)\Big) \\
&= \underbrace{-\frac{1}{|V|}\sum_{v \in V}\frac{1}{|N(v)|}\sum_{u \in N(v)} f_M(v)^\top f_G(u)/\tau}_{\mathcal{L}_{pos}} + \underbrace{\frac{1}{|V|}\sum_{v \in V}\log \sum_{v^- \in V} \exp(f_M(v)^\top f_G(v^-)/\tau)}_{\mathcal{L}_{neg}}.
\end{aligned} \tag{8}$$

Then, we consider the unfolded iterations of descent steps on MLP representation $f_M(v)$. Specifically, we first consider taking the derivatives of $\mathcal{L}_{pos}$ and $\mathcal{L}_{neg}$ on $f_M(v)$:

$$\frac{\partial \mathcal{L}_{pos}}{\partial f_M(v)} = -\frac{1}{|V|}\frac{1}{|N(v)|}\sum_{u \in N(v)} f_G(u)/\tau; \quad \frac{\partial \mathcal{L}_{neg}}{\partial f_M(v)} = \frac{1}{|V|}\sum_{v' \in V}\frac{\exp(f_M(v)^\top f_G(v')/\tau) f_G(v')/\tau}{\sum_{v'' \in V}\exp(f_M(v)^\top f_G(v'')/\tau)}. \tag{9}$$

As we denote representation matrix M as with $f_M(v)$ as the v-th row, the gradients in the Equation (9) can be written as the following matrix forms for simplicity and clarity:

$$\frac{\partial \mathcal{L}_{pos}}{\partial M} = -\frac{1}{|V|}\frac{1}{\tau}\tilde{D}\tilde{A}G; \quad \frac{\partial \mathcal{L}_{neg}}{\partial M} = \frac{1}{|V|}\frac{1}{\tau}D^{-1}AG; \tag{10}$$

where $A = \exp(MG^\top/\tau)$ is the affinity matrix based on feature similarity. $D = \deg(A)$ is the diagonal matrix, whose element in the v-th row and v-th column is the sum of the v-th row of A. In order to reduce the losses on positive pairs and negative pairs, we take a step by performing a gradient descent, which is to update representations M as follows:

$$M^{(t+1)} = M^{(t)} - \eta\frac{\partial(\mathcal{L}_{pos} + \mathcal{L}_{neg})}{\partial M} = M^{(t)} - \frac{\eta}{\tau}(D^{-1}A - \tilde{D}\tilde{A})G; \tag{11}$$

where $M^{(t+1)}$ and $M^{(t)}$ denote the representations before and after the update, respectively, and $\eta > 0$ is the step size of the gradient descent. Note that the constant $1/|V|$ has been absorbed in η. We can easily notice the updating in Equation (11) reveals the global minimum of the learning: $D^{-1}A = \tilde{D}\tilde{A}$. Combining this with Equation (8) completes the proof. □

B.2. Proofs of Theorem 4.2

To prove Theorem 4.2, we first present the following lemma:

Lemma B.1. (Theorem B.3 (page 32) in (HaoChen et al., 2021)). Let f be a minimizer of the spectral contrastive loss: $\mathcal{L}_{SCL} = \sum_{x,x' \in X} -2 \cdot w_{xx'} \cdot f(x)^\top f(x') + w_x w_{x'} \cdot (f(x)^\top f(x'))^2$, where $w_{x,x'} = w(x)w(x'|x)$ is the probability of a random positive pair being (x, x') while w_x is the probability of a randomly selected data point being x, we have:

$$\mathcal{E}(f) \triangleq \min_W \sum_{x \in X} w_x \cdot \mathbb{1}[g_{f,W}(x) \neq y(x)] \leq \frac{\phi^{\hat{y}}}{\lambda_{k+1}}; \tag{12}$$

where $\phi^{\hat{y}} = \sum_{x,x' \in X} w_{xx'} \cdot \mathbb{1}[\hat{y}(x) \neq \hat{y}(x')]$ and W is the downstream linear classifier. λ_{k+1} is the $(K+1)$-th smallest eigenvalues of the matrix $I - D_P^{-1/2} P D_P^{-1/2}$ where $P \in \mathbb{R}^{N \times N}$ is a symmetric probability matrix with $P_{xx'} = w_{xx'}$ and $D_P \in \mathbb{R}^{N \times N}$ is a diagonal matrix with $(D_P)_{xx} = w_x$.

We also introduce the following Lemma in (HaoChen et al., 2021) which asserts that multiplying the embedding matrix on the right by an invertible matrix does not affect the linear probing error.

Lemma B.2. (Lemma 3.1 (page 8) in (HaoChen et al., 2021)). Consider an embedding matrix $F \in \mathbb{R}^{N \times k}$ and a linear classifier $B \in \mathbb{R}^{k \times r}$. Let $D \in \mathbb{R}^{N \times N}$ be a diagonal matrix with positive diagonal entries and $Q \in \mathbb{R}^{k \times k}$ be an invertible matrix. Then, for any matrix $\tilde{F} = D \cdot F \cdot Q$, the linear classifier $\tilde{B} = Q^{-1}B$ on $\tilde{F}$ has the same prediction as B on F. Thus, we have $\mathcal{E}(F) = \mathcal{E}(\tilde{F})$.

Intuitively, this Lemma suggests that although there might not be a single unique optimal solution, when we employ the representation within the context of linear probing, the linear classifier can efficiently handle variations caused by affine transformations. Thus, it produces identical classification errors across different variants when operating in optimal settings.

Theorem 4.2. Let f_M be the global minimum of generalized contrastive loss $\mathcal{L}(\cdot)$ in Equation (4) and $y(v)$ denote the label of v. $\lambda_1 \geq \cdots \geq \lambda_N$ are the eigenvalues with descending order of the normalized adjacency matrix $D^{-1}A$. Then, the linear probing error of f_M is upper-bounded by:

$$\mathcal{E}(f_M) \triangleq \min_W \frac{1}{|V|}\sum_{v \in V} \mathbb{1}[g_{f,W}(v) \neq y(v)] \leq \frac{1-\phi}{1-\lambda_{K+1}}; \tag{13}$$

where $\phi = \frac{1}{|V|}\sum_{v \in V}\frac{1}{|N(v)|}\sum_{u \in N(v)} \mathbb{1}[y(v) = y(u)]$ and K is the dimension of the representation.

Proof. Based on Equation (8), we further have the following:

$$
\begin{aligned}
\mathcal{L}_{ECL} &\approx -\frac{1}{|V|}\sum_{v\in V}\Big(\frac{1}{|N(v)|}\sum_{u\in N(v)} f_M(v)^\top f_G(u)/\tau + \log\sum_{v^-\in V}\exp(f_M(v)^\top f_G(v^-)/\tau)\Big) \\
&= -\frac{1}{|V|}\sum_{v\in V}\Big(\frac{1}{|N(v)|}\sum_{u\in N(v)} f_M(v)^\top f_G(u)/\tau + \log\sum_{v^-\in V}\frac{\exp(f_M(v)^\top f_G(v^-)/\tau)}{|V|}|V|\Big) \\
&= -\frac{1}{|V|}\sum_{v\in V}\Big(\frac{1}{|N(v)|}\sum_{u\in N(v)} f_M(v)^\top f_G(u)/\tau + \log\sum_{v^-\in V}\frac{\exp(f_M(v)^\top f_G(v^-)/\tau)}{|V|} + \log|V|\Big) \\
&\overset{c}{=} -\frac{1}{|V|}\sum_{v\in V}\Big(\frac{1}{|N(v)|}\sum_{u\in N(v)} f_M(v)^\top f_G(u)/\tau + \log\sum_{v^-\in V}\frac{\exp(f_M(v)^\top f_G(v^-)/\tau)}{|V|}\Big) \\
&\leq -\frac{1}{|V|}\sum_{v\in V}\Big(\frac{1}{|N(v)|}\sum_{u\in N(v)} f_M(v)^\top f_G(u)/\tau + \sum_{v^-\in V}\frac{1}{|V|}(f_M(v)^\top f_G(v^-)/\tau)\Big) \quad (14)\\
&= -\frac{1}{|V|}\sum_{v\in V}\frac{1}{|N(v)|}\sum_{u\in N(v)} f_M(v)^\top f_G(u)/\tau + \frac{1}{|V|}\frac{1}{|V|}\sum_{v\in V}\sum_{v^-\in V} f_M(v)^\top f_G(v^-)/\tau \\
&\leq -\frac{1}{|V|}\sum_{v\in V}\frac{1}{|N(v)|}\sum_{u\in N(v)} f_M(v)^\top f_G(u)/\tau + \frac{1}{|V|}\frac{1}{|V|}\sum_{v\in V}\sum_{v^-\in V}(f_M(v)^\top f_G(v^-))^2/\tau \quad (15)\\
&\overset{c}{=} -\frac{1}{|V|}\sum_{v\in V}\frac{1}{|N(v)|}\sum_{u\in N(v)} 2f_M(v)^\top f_G(u) + \frac{1}{|V|}\frac{1}{|V|}\sum_{v\in V}\sum_{v^-\in V}(f_M(v)^\top f_G(v^-))^2 \triangleq \mathcal{L}_{cross}; \quad (16)
\end{aligned}
$$

where the symbol $\overset{c}{=}$ indicates equality up to a multiplicative and/or additive constant. Here, we utilize Jensen's inequality in (14). Inequality (15) holds because $f_M(v)$ and $f_G(u)$ are ℓ_2 normalized, and we assume the embedding heads consisting of last-layer ReLU neural networks. We define the two metrics $\tilde{M}$ and $\tilde{G}$. Equation (16) holds if we set the temperature of positive pairs is twice to it of negative pairs. Here $(\tilde{M})_v = |V|^{-1/2} f_M(v)$ and $(\tilde{G})_u = |V|^{-1/2} f_G(u)$. Then, the loss in Equation (16) is equivalent to the low-rank asymmetric matrix factorization loss up to a constant:

$$\mathcal{L}_{AMF} = \|D^{-1}A - \tilde{M}\tilde{G}^\top\| = \mathcal{L}_{cross} + \text{const} \tag{17}$$

According to Eckart–Young–Mirsky theorem (Eckart & Young, 1936), the optimal solution $\tilde{M}^*$ and $\tilde{G}^*$ of $\mathcal{L}_{AMF}$ can be respectively represented as follows:

$$\tilde{M}^*(\tilde{G}^*)^\top = U^K \operatorname{deg}(\sigma_1, \ldots, \sigma_K)(V^K)^\top \tag{18}$$

where we denote $D^{-1}A = U\Sigma V^\top$ as the spectral decomposition of $D^{-1}A$. $(\sigma_1, \ldots, \sigma_K)$ are the K-largest eigenvalue of $D^{-1}A$. The k-th column of $U^K \in \mathbb{R}^{|V|\times K}$ is the corresponding eigenvector of the k-th largest eigenvalue and $V^K \in \mathbb{R}^{|V|\times K}$ is a unitary matrix. Then the optimal solution $\tilde{M}^*$ and $\tilde{G}^*$ can be represented as follows:

$$\tilde{M}^* = U^K BR; \quad \tilde{G}^* = V^K \operatorname{deg}(\sigma_1, \ldots, \sigma_K)B^{-1}R; \tag{19}$$

where $R \in \mathbb{R}^{K\times K}$ is a unitary matrix and B is an invertible diagonal matrix. Since $(\tilde{M})_v = |V|^{-1/2} f_M(v)$ and $(\tilde{G})_u = |V|^{-1/2} f_G(u)$, we have:

$$f_M(v) = |V|^{1/2}((U^K)_v BR)^\top; \quad f_G(u) = |V|^{1/2}((V^K)_u \operatorname{deg}(\sigma_1, \ldots, \sigma_K)B^{-1}R)^\top. \tag{20}$$

Similar, if we consider optimizing following uni-model spectral contrastive loss:

$$\mathcal{L}_{uni} = -\frac{1}{|V|}\sum_{v\in V}\frac{1}{|N(v)|}\sum_{u\in N(v)} 2f_M(v)^\top f_M(u) + \frac{1}{|V|}\frac{1}{|V|}\sum_{v\in V}\sum_{v^-\in V}(f_M(v)^\top f_M(v^-))^2. \tag{21}$$

The optimal solution $\hat{f}_M$ of this uni-model spectral contrastive loss can be represented as follow:

$$\hat{f}_M(v) = |V|^{1/2}((U^K_{uni})_v B_{uni} R_{uni})^\top. \tag{22}$$

Since the uni-model spectral contrastive loss in Equation (21) also decomposes the matrix $\mathsf{D}^{-1}\mathsf{A}$, the $\mathsf{U}^K_{uni} = \mathsf{U}^K$. As B, R, B_{uni}, R_{uni} are invertible matrices, and the product of the invertible matrices is still invertible, we have the following:

$$f_M(v) = \hat{f}_M(v)\mathsf{T}, \tag{23}$$

where $\mathsf{T} = (\mathsf{B}_{uni})^{-1}(\mathsf{R}_{uni})^{-1}\mathsf{B}\mathsf{R}$. With Lemma B.2, we establish that $E(f_M) = E(\hat{f}_M)$. Additionally, we observe that the loss in Equation (21) shares the same form as the spectral contrastive loss when we define $\frac{1}{|V|}\mathsf{D}^{-1}\mathsf{A} = \hat{\mathsf{A}}$ i.e., $w_x = \frac{1}{|V|}$ and $w_{x'x} = (\mathsf{D}^{-1}\mathsf{A})_{x,x'}$. It's worth noting that $\mathsf{D}^{-1}\mathsf{A} = \mathsf{D}^{-1/2}\mathsf{A}\mathsf{D}^{-1/2}$ forms a symmetric matrix due to our random sampling process, which ensures that the same neighbors are sampled for each central node, approximately resulting in equal node degrees. Thus, with Lemma B.1, we can obtain the following:

$$E(f_M) = E(\hat{f}_M) \lesssim \min_W \frac{1}{|V|}\sum_{v\in V}[g_{f,W}(v) \neq y(v)] \cdot \frac{1-\alpha}{1-\sigma_{K+1}} \tag{24}$$

where $\alpha = 1/|V|\sum_{v\in V}\frac{1}{|N(v)|}\sum_{u\in N(v)}[y(v) = y(u)]$ and σ_{K+1} is the $(K+1)$-th largest singular value of the normalized adjacency matrix $\mathsf{D}^{-1}\mathsf{A}$. Given the above, the proof is finished. □

C. Experimental Details

Table 7. Statistics of Datasets.

	Cora	Citeseer	Pubmed	Photo	Flickr	WikiCS	Actor	Wisconsin	Cornell	Texas	snap-patents	ogbn-arxiv	ogbn-papers100M
#Nodes	2,708	3,327	19,717	7,650	89,250	11,701	7,600	251	183	183	2,923,922	169,343	111,059,956
#Edges	5,278	4,552	44,324	119,081	899,756	216,123	33,544	466	295	309	13,975,788	1,166,243	1,615,685,872
#Classes	7	6	3	8	7	10	5	5	5	5	5	40	172
$H(G)$	0.83	0.71	0.79	0.85	0.32	0.65	0.22	0.16	0.11	0.06	0.22	0.66	0.54
$S(G)$	0.89	0.81	0.87	0.91	0.33	0.75	0.68	0.42	0.40	0.79	0.29	0.79	0.71

C.1. One-hop Node Homophily Level

We use the node homophily ratio to measure the one-hop neighbor homophily of the graph (Pei et al., 2019). Specifically, the node homophily ratio $H(G)$ can be computed as follows:

$$H(G) = \frac{1}{|V|}\sum_{v\in V}\frac{1}{N(v)}\sum_{u\in N(v)}(y(v) = y(u)). \tag{25}$$

C.2. One-hop Neighborhood Context Similarity

To validate the assumption that nodes belonging to an identical semantic category are likely to exhibit similar patterns in their one-hop neighborhoods, even in heterophilic graphs, we examine whether nodes with the same label demonstrate similar distributions of labels in their neighborhoods regardless of homophily. We evaluate this characteristic by computing the class neighborhood similarity (Ma et al., 2021), which is defined as:

$$s(m, m') = \frac{1}{|V_m||V_{m'}|}\sum_{u\in V_m, v\in V_{m'}}\cos(d(u), d(v)), \tag{26}$$

where M denotes the total number of classes, V_m represents the set of nodes classified as m, and $d(u)$ is the empirical histogram of the labels of node u's neighbors across M classes. The cosine similarity function is represented by $\cos(\cdot)$. This metric for cross-class neighborhood similarity quantifies the differences in neighborhood distributions between varying classes. When $m = m', s(m, m')$ determines the intra-class similarity. To quantify the neighborhood similarity, we take the average of the intra-class similarities across all classes:

$$S(G) = \sum_{m=1}^{M}\frac{1}{M}s(m, m). \tag{27}$$

If nodes with identical labels exhibit similar neighborhood distributions, then the class neighborhood similarity $S(G)$ will be high. Table 7 shows that heterophilic graphs exhibit stronger neighborhood similarity, even when the homophily ratio is low.

C.3. Datasets Details

The statistics of the benchmark datasets, including homophily levels and 1-hop neighborhood similarities, are given in Table 7. All datasets and public splits can found in PyTorch Geometric: `https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html`.

Cora, Citeseer, and Pubmed. (Yang et al., 2016) These datasets serve as some of the most prevalent benchmarks for node classification. Each one constitutes a graph representing citations, with nodes symbolizing documents and edges depicting citation relationships between them. The classification of each node is determined by the respective research field. Features of the nodes are derived from a bag-of-words model applied to their abstracts. We utilize the public split: a fixed 20 nodes from each class for training and another distinct set of 500 and 1,000 nodes for validation and testing, respectively.

WikiCS. (Mernyei & Cangea, 2020) This graph consists of nodes corresponding to Computer Science articles, with edges based on hyperlinks and 10 classes representing different branches of the field. We adopt a 10/10/80% training/validation/testing public split provided by PyTorch Geometric.

Photo. (McAuley et al., 2015) This graph originates from the Amazon co-purchase graph (McAuley et al., 2015), where nodes denote products and edges connect pairs of items often bought together. In the Photo dataset, products are categorized into eight classes based on their category, and the node features are represented by a bag-of-words model of the product's reviews. We employ a public split of the nodes into training, validation, and testing sets, following a 10/10/80% ratio as described in (Thakoor et al., 2021).

Flickr. (Zeng et al., 2020) In this graph, each node symbolizes an individual image uploaded to Flickr. An edge is established between the nodes of two images if they share certain attributes, such as geographic location, gallery, or user comments. The node features are represented by a 500-dimensional bag-of-words model provided by NUS-WIDE. Regarding labels, we examined the 81 tags assigned to each image and manually consolidated them into 7 distinct classes, with each image falling into one of these categories. We use a random node division method, adhering to a 50/25/25% split for training, validation, and testing sets, following (Zeng et al., 2020).

Cornell, Wisconsin and Texas. (Pei et al., 2019) These are networks of webpages gathered from the computer science departments of various universities by Carnegie Mellon University. In each network, the nodes represent individual webpages, while the edges signify hyperlinks between them. The features of the nodes are depicted using bag-of-words representations of the webpages. The objective is to categorize each node into one of five classes.

Actor. (Pei et al., 2019) This is a subgraph induced solely by actors, derived from the broader film-director-actor-writer network. In this subgraph, nodes represent actors, while edges denote the co-occurrence of two nodes on the same Wikipedia page. The features of the nodes are constituted by keywords found on Wikipedia pages. Labels are categorized into five groups based on the content of the actor's corresponding Wikipedia page.

For **Texas, Wisconsin, Cornell, and Actor**, we use the raw data provided by Geom-GCN (Pei et al., 2019) with the standard fixed 10-fold split for our experiment. In addition to the above graphs, we also conduct experiments on the following three large-scale graphs: snap-patents, ogbn-arxiv and ogbn-papers100M,

Snap-patents. (Lim et al., 2021) The Snap-patents dataset encompasses a collection of utility patents from the United States, where each node represents a patent, and edges are formed between patents that cite one another. The features of the nodes are extracted from the metadata of the patents. In this work, we introduce a task aiming to predict the time at which a patent was granted, which is categorized into five classes. We used the unprocessed data from (Lim et al., 2021), employing the standard 10-fold split for our experimental setup.

Ogbn-arxiv and Ogbn-papers100M. (Hu et al., 2020) These two large-scale datasets are collected by Hu et al. (2020). Ogbn-arxiv and Ogbn-papers100M are citation networks where each node represents a paper. The corresponding features consist of titles and abstracts, and node labels are the primary categories of the papers (Chien et al., 2021). We used the public split ratio provided in the OGB benchmark.

C.4. Transductive and Inductive Settings for Unsupervised Representation Learning

Transductive Setting. To fully evaluate the model, we consider two settings: transductive (tran) and inductive (ind). In the transductive setting, our evaluation consists of two phases. Initially, we pre-train models on graph G, followed by the generation of representations for all nodes within the graph, denoted as z_v for $v \in V$. Subsequently, we employ a linear

Table 8. Graph classification results (%) on MUTAG and PROTEINS

Method	Graph-MLP	VGAE	CCA-SSG	BGRL	GraphECL
MUTAG	75.8±2.0	84.4±0.6	85.8±0.4	86.8±1.3	**88.5±1.2**
PROTEINS	71.1±1.5	74.0±0.5	73.1±0.6	73.5±0.7	**75.2±0.3**

classifier trained in fixed learned representations using labeled data Z^L and Y^L. Finally, we assess the remaining inferred representations Z^U with the corresponding labels Y^U.

Inductive Setting. In the unsupervised inductive setting, we randomly select 20% of the nodes as a test set for inductive evaluation. Specifically, we partition the unlabeled nodes V^U into two separate subsets: observed and inductive (i.e., $V^U = V^U_{\text{obs}} \cup V^U_{\text{ind}}$). This leads to the creation of three distinct graphs: $G = G^L \cup G^U_{\text{obs}} \cup G^U_{\text{ind}}$, where no nodes are shared between $G^L \cup G^U_{\text{obs}}$ and G^U_{ind}. Importantly, during training, we remove the edges that connect $G^L \cup G^U_{\text{obs}}$ and G^U_{ind}. Upon completing the self-supervised pretraining in $G^L \cup G^U_{\text{obs}}$, we generate representations for all nodes. Consequently, the learned representations and associated labels are partitioned into three separate sets: $Z = Z^L \cup Z^U_{obs} \cup Z^U_{\text{ind}}$ and $Y = Y^L \cup Y^U_{obs} \cup Y^U_{\text{ind}}$. A downstream classifier is then trained on the learned representations Z^L and lables Y^L. Finally, we evaluate the remaining representations Z^U and Z^U_{ind} in the downstream classifier with labels Y^U_{obs} and Y^U_{ind}, respectively.

C.5. Setup and Hyper-parameter Settings

We utilized the official implementations publicly released by the authors for the baselines. To ensure a fair comparison, we conducted a grid search to determine the optimal hyperparameters. Our experiments were conducted on a machine equipped with NVIDIA RTX A100 GPUs with 80GB memory. For all experiments, we employed the Adam optimizer (Kingma & Ba, 2014). A small-scale grid search was used to select the best hyperparameters for all methods. Specifically, for our approach, we explored the following hyperparameter ranges: from $\{0.001, 0.01, 0.1, 0.5, 1\}$, K from $\{256, 512, 1024, 2048, 4096\}$, from $\{0.5, 0.75, 0.99, 1\}$, and the number of negative pairs M from $\{1, 5, 10\}$ when negative sampling was used. Furthermore, we tuned the learning rate from the set $\{$1e-3, 5e-3, 1e-4$\}$ and the weight decay from the set $\{$0, 1e-4, 3e-4, 1e-6$\}$. The selection of the optimal hyperparameter configuration was based on the accuracy on the validation set.

D. Additional Experimental Results

D.1. Graph Classification Performance

For the graph classification task, we can use a non-parameterized graph pooling (readout) function, such as MeanPooling, to obtain the graph-level representation. In our experiments, we focus on graph classification using two benchmarks: PROTEINS and MUTAG. We follow the same experimental setup as for GraphCL (You et al., 2020). The results are presented in Table 8. From the table, we observe that our GraphECL performs well on the graph classification task and achieves better performance compared to the baselines. This observation, coupled with the node classification results, underscores the effectiveness of GraphECL in acquiring more expressive and resilient node representations for a variety of downstream tasks. These findings further validate that modeling one-hop neighborhood patterns confers advantages on downstream tasks on real-world graphs with varying degrees of homophily.

D.2. Performance on Long Range Graph Benchmark

To further evaluate the effectiveness of GraphECL in capturing inter-neighborhood information, we also evaluate it on Long Range Graph Benchmark (Dwivedi et al., 2022). Specifically, we compare GraphECL with two graph contrastive learning methods, BGRL and CCA-SSG, on PascalVOC-SP (Dwivedi et al., 2022) in Table 9. For all methods, we employ the GCN as the backbone. From the table below, we can observe that GraphECL performs well in PascalVOC-SP, achieving better performance compared to the baselines. This further strengthens GraphECL on capturing inter-neighborhood information.

D.3. The Effect of Size of Negative Pairs

We conducted a sweep over the size of negative samples, denoted as M, to study its impact on performance. We varied M across the values $1, 5, 10$. For each value M, we first learned node representations and subsequently applied these learned representations to node classification. The results of this experiment are shown in Figure 9. From the figure, we observe

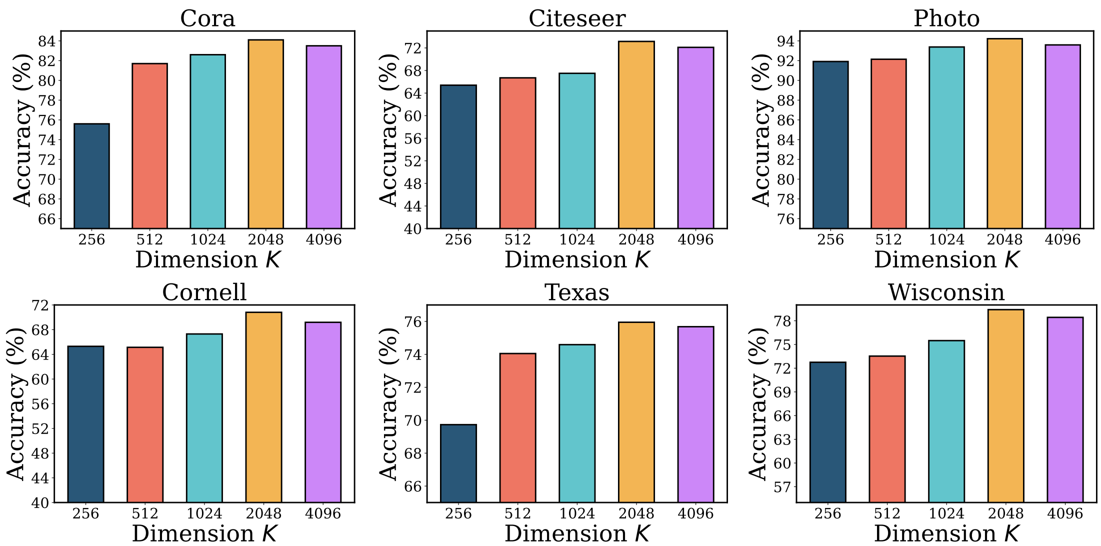

Figure 7. The effect of the dimensions of representations.

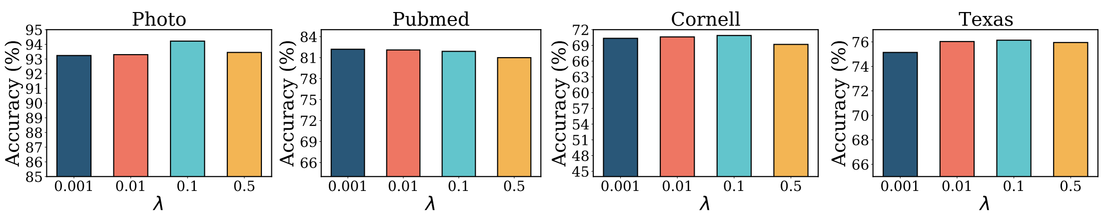

Figure 8. The effect of the hyperparameter λ.

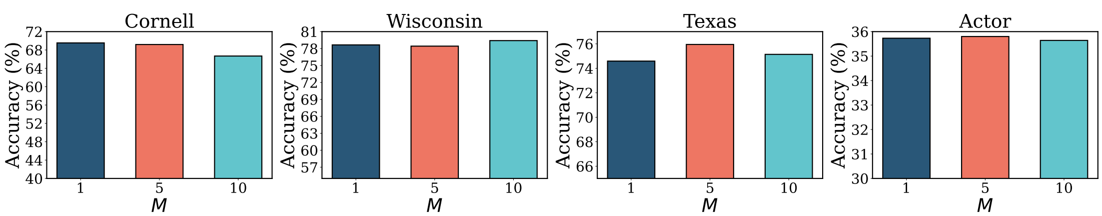

Figure 9. The effect of the size of negative pairs.

that even a small number of negative samples, such as $M = 5$, is sufficient to achieve good performance across all graphs, demonstrating that `GraphECL` is particularly robust to reduced negative pairs.

D.4. The Effect of Representation Dimension

We investigate the impact of different dimensions of representations. Figures 7 show the results of node classification with varying dimensions on homophilic and heterophilic graphs. From the figure, we can observe that larger dimensions often yield better results for both homophilic and heterophilic graphs. This observation is consistent with Theorem 4.2, which shows that a larger dimension can effectively reduce the upper bound of downstream errors. Training with extremely large dimensions for some graphs may lead to a slight drop of performance, as `GraphECL` may suffer from the over-fitting issue.

Table 9. The performance on the long-range graph benchmark PascalVOC-SP.

Method	BGRL	CCA-SSG	GraphECL
PascalVOC-SP	0.1356±0.0087	0.1437±0.0095	**0.1588±0.0091**