Bootstrap Latents of Nodes and Neighbors for Graph Self-Supervised Learning

Yunhui Liu[1,2], Huaisong Zhang[3], Tieke He[1,2] (✉), Tao Zheng[1,2], and Jianhua Zhao[1,2]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
{lyhcloudy1225,hetieke}@gmail.com
[2] Software Institute, Nanjing University, Nanjing, China
[3] Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China

Abstract. Contrastive learning is a significant paradigm in graph self-supervised learning. However, it requires negative samples to prevent model collapse and learn discriminative representations. These negative samples inevitably lead to heavy computation, memory overhead and class collision, compromising the representation learning. Recent studies present that methods obviating negative samples can attain competitive performance and scalability enhancements, exemplified by bootstrapped graph latents (BGRL). However, BGRL neglects the inherent graph homophily, which provides valuable insights into underlying positive pairs. Our motivation arises from the observation that subtly introducing a few ground-truth positive pairs significantly improves BGRL. Although we can't obtain ground-truth positive pairs without labels under the self-supervised setting, edges in the graph can reflect noisy positive pairs, i.e., neighboring nodes often share the same label. Therefore, we propose to expand the positive pair set with node-neighbor pairs. Subsequently, we introduce a cross-attention module to predict the supportiveness score of a neighbor with respect to the anchor node. This score quantifies the positive support from each neighboring node, and is encoded into the training objective. Consequently, our method mitigates class collision from negative and noisy positive samples, concurrently enhancing intra-class compactness. Extensive experiments are conducted on five benchmark datasets and three downstream task node classification, node clustering, and node similarity search. The results demonstrate that our method generates node representations with enhanced intra-class compactness and achieves state-of-the-art performance. Our implementation code is available at https://github.com/Cloudy1225/BLNN.

Keywords: Self-Supervised Learning · Graph Representation Learning · Graph Neural Networks

1 Introduction

Graph self-supervised learning (GSSL) is a promising paradigm for learning more informative representations without human annotations. Typically, GSSL models are pre-trained using well-designed pretext objectives, which serve as effective initializations for diverse downstream tasks [19]. Consequently, GSSL has made substantial advancements in graph representation learning. It offers performance, generalizability,

and robustness metrics comparable to or even surpassing those of supervised methods [30,28,2].

A major branch of GSSL is graph contrastive learning (GCL) methods [41,42], which aim to learn representations by maximizing the agreement between two augmented samples (positive pair) while minimizing the similarities with other samples (negative pairs). The constructed negative pairs is crucial for preventing model collapse and generating discriminative representations [32]. Consequently, current GCL methods inherently rely on increasing the quantity and quality of negative samples. This reliance not only introduces additional computational and memory costs but also leads to the class collision issue, where different samples from the same class are erroneously considered negative pairs, thereby impeding representation learning for classification [25]. To address these issues, recent non-contrastive methods have explored the prospect of learning without negative samples [37,28,1,15,17,27]. Among these methods, Bootstrapped Graph Latents (BGRL) [28], derived from BYOL [7], has achieved competitive performance and heightened scalability. BGRL learns node representations by using representations of one augmented view to predict another view, i.e., maximizing the similarity between the prediction and its paired target. Simultaneously, BGRL strategically leverages the asymmetry between the online branch (with gradient) and the target branch (without gradient) to alleviate model collapse.

However, BGRL fails to account for inherent graph homophily, which indicates the phenomenon that neighboring nodes tend to share the same semantic label and thus offers valuable insights into underlying positive pairs. *Why does exploiting the homophily pattern make sense?* In practice, some supervised metric learning methods [13,36,34], which employ architectures and objectives akin to self-supervised learning, have illustrated that introducing more ground-truth positive pairs (i.e., samples with the same label) significantly enhances representation learning for classification. Such success inspires us that mining potential positive pairs could empower the model to learn highly intra-class-compacted representations, which are more conducive to classification. Our hypothesis is validated through empirical studies in Section 4.1. Unfortunately, unlike the supervised setting, obtaining ground-truth positive pairs is unfeasible due to the absence of labels under the self-supervised setting. But fortunately, the homophily pattern is evident in various real-world graphs [21], where neighboring nodes can be seen as noisy positive pairs. Consequently, exploiting such neighbor information holds promise for graph self-supervised learning.

Based on the above analysis, we propose Bootstrap Latents of Nodes and Neighbors (BLNN) to enhance Bootstrapped Graph Latents by incorporating neighbor information. Specifically, we first expand the positive pair set with node-neighbor pairs based on the graph homophily pattern. However, although connected nodes tend to share the same label in the homophily scenario, there also exist inter-class edges, especially near the decision boundary between two classes. Treating these inter-class connected nodes as positive (i.e., false positive) pairs would inevitably compromise overall performance. To alleviate this class collision caused by false positive pairs, we further introduce an attention module to compute a supportiveness score of each neighbor representation with respect to the current view anchor node. This score serves as a soft measure of the supportiveness associated with each neighbor contributing to the current anchor node

during loss computations. Basically, a higher supportiveness often stands for a higher weight to intra-class node-neighbor pairs. To this end, our BLNN incorporates soft positive node-neighbor pairs to support the anchor node for loss computations, resulting in more intra-class-compacted and discriminative node representations. The contributions of our work can be summarized as follows:

- We empirically demonstrate the efficacy of introducing more ground-truth positive pairs in boosting the negative-sample-free method BGRL. And we propose exploiting the graph homophily to mining positive pairs in the absence of labels.
- We expand the positive pair set with node-neighbor pairs and propose a cross-attention module to weight the contribution of each neighbor to loss computations. This approach mitigates class collision resulting from false positive node-neighbor pairs.
- Extensive experiments are conducted on five benchmark datasets and three downstream task node classification, node clustering, and node similarity search. The results demonstrate that our method generates node representations with enhanced intra-class compactness and achieves state-of-the-art performance.

2 Related Work

2.1 Graph Self-Supervised Learning

Recently, numerous research efforts have been devoted to graph self-supervised learning, and a branch based on multi-view learning has garnered attention owing to its superior performance. The basic idea involves ensuring consensus among multiple views derived from the same sample under different graph transformations to optimize model parameters [19]. A crucial aspect of these methods is the prevention of trivial solutions, where all representations converge either to a constant point (i.e., complete collapse) or to a subspace (i.e., dimensional collapse). The existing methods can be broadly classified into two groups: contrastive and non-contrastive approaches, each delineated by its strategy for mitigating model collapse.

Contrastive-based methods typically follow the criterion of mutual information maximization [10], whose objective functions involve contrasting positive pairs with negative ones. Pioneering works, such as DGI [30] and GMI [24], focus on unsupervised representation learning by maximizing mutual information between node-level representations and a graph summary vector, employing the Jensen-Shannon estimator [23]. MVGRL [9] proposes to learn both node-level and graph-level representations by performing node diffusion and contrasting node representations to augmented graph representation. GRACE [41] and its variants GCA [42], gCooL [16], CSGCL [2] learn node representations by pulling together the representations of the same node in two augmented views while pushing away the representations of the other nodes in two views [32]. Despite the success of contrastive learning on graphs, they require a large number of negative samples with carefully crafted encoders and augmentation techniques to learn discriminative representations, making them suffer seriously from heavy computation, memory overhead and class collision [25].

Non-contrastive methods discard negative samples, necessitating specialized strategies to avoid collapsed solutions. CCA-SSG [37], G-BT [1] and iGCL [17] learn augmentation invariant information while introducing feature decorrelation to capture orthogonal features and prevent dimensional collapse. BGRL [28], derived from BYOL [7], introduces an online network along with a target network, where the target network is updated with a moving average of the online network to avoid collapse. AFGRL [15] identifies nodes as positive samples by considering both local structural information and global graph semantics, sidestepping the need for an augmented graph view and negative sampling. SGCL [27] uncovers the hidden factors contributing to BGRL's success and simplifies the architecture design. In this paper, we propose mining potential positive pairs from neighboring nodes to enhance BGRL.

2.2 Generation of Positive and Negative Pairs

There are two common approaches to generating positive and negative pairs, depending on the availability of label information. In the supervised setting, where label information is available, positive pairs consist of samples within the same class, while negative pairs comprise samples from different classes [13,36,34]. In the self-supervised setting without label information, a typical strategy is to generate different views of the original sample via augmentation [12]. Here, two views of the same sample serve as positive pairs for each other, while those of different samples serve as negative pairs. However, such instance discrimination based methods inevitably a class collision issue, which means even for very similar samples, they still need to be pushed apart.

To mitigate the class collision issue, some studies focus on mining positive pairs from nearest neighbors [40,3,5,15] while some propose methods without negative pairs [7,37,28,15]. In the domain of graph, AF-GCL [31] regards multi-hop neighboring nodes as potential positive pairs, utilizing well-designed similarity metrics to identify the most similar nodes as positive pairs; nevertheless, this method still necessitates a considerable number of negative pairs. AFGRL [15] and HomoGCL [18] identify positive pairs by considering the local structural information and the global semantics of graphs, but they require performing time-consuming K-means clustering on the entire set of node representations to capture global semantic information. Our BLNN differs from previous work in the following three highlights: 1) BLNN, derived from BGRL [28], is a non-contrastive method, eliminating the introduction of class collision arising from false negative pairs. 2) BLNN treats all one-hop node-neighbor pairs as candidate positive pairs, simplifying the selection of candidate neighbors from the K-NN search. 3) BLNN employs a cross-attention module, instead of the time-consuming K-means, to mitigate class collision caused by noisy positive node-neighbor pairs.

3 Preliminary

3.1 Problem Statement

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represenst an attributed graph, where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denote the node set and the edge set, respectively. The graph $\mathcal{G}$ is associated

with a feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, where $\boldsymbol{x}_i \in \mathbb{R}^p$ represents the feature of v_i, and an adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$, where $\boldsymbol{A}_{i,j} = 1$ if and only if $(v_i, v_j) \in \mathcal{E}$. During training in the self-supervised setting, no task-specific labels are provided for $\mathcal{G}$. The primary objective is to learn an embedding function $f_\theta(\boldsymbol{A}, \boldsymbol{X})$ that transforms $\boldsymbol{X}$ to $\boldsymbol{H}$, where $\boldsymbol{H} \in \mathbb{R}^{n \times d}$ and $d \ll p$. The pre-trained representations are intended to encapsulate both attribute and structure information inherent in $\mathcal{G}$ and can be easily transferable to various downstream tasks such as node classification, node clustering, and node similarity search.

3.2 Graph Homophily

Graph homophily suggests that neighboring nodes often belong to the same class, offering valuable prior knowledge in real-world graphs such as citation networks, co-purchase networks, or friendship networks [21]. A well-used metric for quantifying graph homophily is edge homophily, which is defined as the fraction of intra-class edges:

$$\mathcal{H} = \frac{1}{|\mathcal{E}|} \sum_{(v_i, v_j) \in \mathcal{E}} \mathbb{I}(y_i = y_j), \tag{1}$$

where y_i denotes the class of v_i and $\mathbb{I}$ represents the indicator function. In Table 1, edge homophily values for five benchmark datasets are presented. The table illustrates that the majority of edges are intra-class, indicating the potential to mine positive pairs from node-neighbor pairs.

3.3 Bootstrapped Graph Latents

We first introduce the pioneer work Bootstrapped Graph Latents (BGRL) [28], which aims to maximize the similarity between representations of the same node generated from two different augmented graph views and employs asymmetric architectures to avoid collapsed representations. BGRL consists of three major components: 1) a random graph augmentation generator $\mathcal{T}$; 2) two asymmetric graph encoders, i.e., the online encoder f_θ and the target encoder f_ϕ; 3) an objective function to maximize the similarity between the positive pair.

Graph View Augmentation. Given the adjacency matrix $\boldsymbol{A}$ and feature matrix $\boldsymbol{X}$ of a graph $\mathcal{G}$, BGRL employs feature masking and edge dropping to enhance both graph attributes and topological information (see Appendix A.3). The augmentation function $\mathcal{T}$ comprises all possible graph transformation operations, and each $t \sim \mathcal{T}$ corresponds to a specific transformation applied to graph $\mathcal{G}$. At each training epoch, BGRL first samples two random augmentation functions $t^1 \sim \mathcal{T}$ and $t^2 \sim \mathcal{T}$, and then generates two views $\mathcal{G}^1 = (\boldsymbol{A}^1, \boldsymbol{X}^1)$ and $\mathcal{G}^2 = (\boldsymbol{A}^2, \boldsymbol{X}^2)$ based on the chosen functions.

Node Representations Generation. Different from the classical contrastive learning frameworks with a shared graph encoder, BGRL employs two asymmetric graph encoders to avoid representation collapse. The online encoder f_θ generates an online representations from the first augmented graph, $\boldsymbol{H}^1 = f_\theta(\boldsymbol{A}^1, \boldsymbol{X}^1)$. Similarly, the target encoder f_ϕ produces a target representation of the second augmented graph, $\boldsymbol{H}^2 = f_\phi(\boldsymbol{A}^2, \boldsymbol{X}^2)$. The online representation is then input into a node-level predictor,

p_θ (implemented as a MLP), which produces a prediction of the target representation, $\boldsymbol{Z}^1 = p_\theta(\boldsymbol{H}^1)$.

Positive Pair Similarity Maximization. The learning process of BGRL centers around maximizing the cosine similarity between the predicted target representations $\boldsymbol{Z}^1$ and the true target representations $\boldsymbol{H}^2$, i.e., positive pairs. The objective function is defined as

$$\mathcal{L}_{BGRL} = -\frac{1}{n}\sum_{i=1}^{n}\frac{\boldsymbol{z}_i^1 \cdot \boldsymbol{h}_i^2}{\| \boldsymbol{z}_i^1 \| \| \boldsymbol{h}_i^2 \|}, \tag{2}$$

where $(\cdot)$ denotes the dot production, and $\| \cdot \|$ represents the ℓ_2 normalization. Notably, only the online encoder parameters θ are updated with respected to the gradients from the objective function while the target encoder parameters ϕ are updated as an exponential moving average (EMA) of θ with a decay rate t, i.e., $\phi = t\phi + (1-t)\theta$. Therefore, BGRL utilizes the outputs from the ensemble-optimized parameters as targets, progressively enhancing the model in a step-by-step fashion, an approach commonly known as bootstrapping.

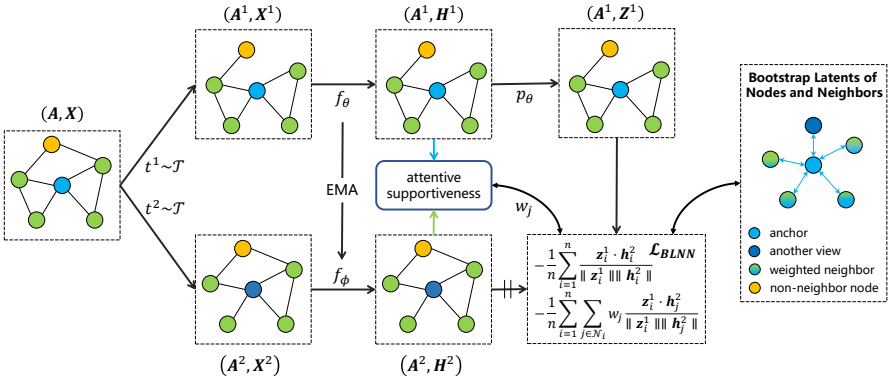


Fig. 1: Overview of our proposed BLNN method. Given a graph, we first generate two different views using augmentations t^1, t^2. From these, we use encoders f_θ, f_ϕ to form online and target node representations $\boldsymbol{H}^1, \boldsymbol{H}^2$. They are then fed into the attention module to compute the supportiveness w_j of the neighbor v_j w.r.t. the anchor node v_i. The predictor p_θ uses $\boldsymbol{H}^1$ to form a prediction $\boldsymbol{Z}^1$ of the target $\boldsymbol{H}^2$. The final objective is computed as a combination of the alignment of node-itself pairs and the supportiveness-weighted alignment of node-neighbor pairs. Note that the alignment is achieved by maximizing the cosine similarity between corresponding rows of $\boldsymbol{Z}^1$ and $\boldsymbol{H}^2$, flowing gradients only through $\boldsymbol{Z}^1$. The target parameters ϕ are updated as an exponentially moving average of θ.

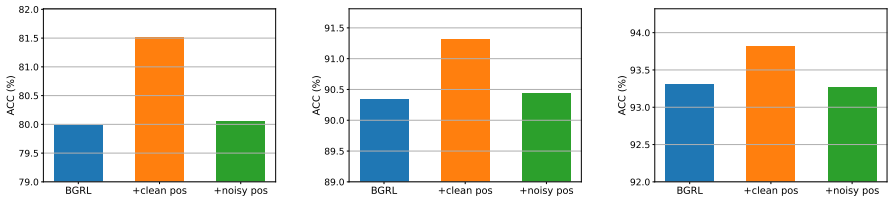
4 Methodology

In this section, we present an overview of the proposed BLNN, as depicted in Figure 1. In Section 4.1, we empirically analyze our motivation to introduce more ground-truth

positive pairs from node-neighbor pairs for graph self-supervised learning. Then, we describe how to mine high-confidence positive information from node-neighbor pairs in Section 4.2.

4.1 Motivation

As discussed in the introduction, some supervised metric learning methods [13,36,34], which employ architectures and objectives similar to self-supervised learning, have shown that introducing more ground-truth positive pairs significantly enhances representation learning for classification. Such success inspires us that mining potential positive pairs could empower BGRL to learn highly intra-class-compacted representations, which are more conducive to classification.



(a) WikiCS (b) Computer (c) CS

Fig. 2: Empirical studies on WikiCS, Computer and CS. “noisy pos" indicates raw node-neighbor pairs in the input graph, while “clean pos" indicates clean node-neighbor pairs that all are intra-class pairs.

Empirical Analysis. To verify our hypothesis, we conduct experiments by incorporating a small subset of the whole ground-truth positive pair set from an oracle perspective and assessing its influence on classification. According to the graph homophily, neighboring nodes often share the same class. Therefore, we first treat all node-neighbor pairs as noisy candidate positive pairs. Subsequently, we manually filter out inter-class pairs, retaining only the intra-class pairs as the clean positive pairs. We then extend the objective function Eq.(2) with an additional alignment of above intra-class node-neighbor pairs to train BGRL. Figure 2 illustrates the results of node classification across three datasets, revealing two key observations: 1) The incorporation of clean positive node-neighbor pairs consistently and significantly improves classification performance. 2) However, simply treating raw node-neighbor pairs as ground-truth positive pairs yields only marginal improvement or even performance degradation, as raw node-neighbor pairs include inter-class pairs, which would cause class collision.

Based on the above observations, we propose to enhance BGRL using two key strategies: 1) expanding the positive pair set with node-neighbor pairs; 2) mitigating class collision caused by false positive node-neighbor pairs via a cross-attention weighting module.

4.2 Bootstrap Latents of Nodes and Neighbors

Motivated by the observations presented in Section 4.1, we introduce Bootstrap Latents of Nodes and Neighbors (BLNN) to enhance Bootstrapped Graph Latents (BGRL). We follow the BGRL framework illustrated in Section 3.3.

Objective Function. Our BLNN first treats node-neighbor pairs as candidate positive pairs, leveraging the neighbor set $\mathcal{N}_i$ to support the anchor node v_i. Subsequently, it introduces an adaptive measurement of supportiveness through a cross-attention module to mitigate class collision resulting from false positive node-neighbor pairs. Specifically, for each neighbor $v_j \in \mathcal{N}_i$, we input its target representation $\boldsymbol{h}_j^2$ and the anchor's online representation $\boldsymbol{h}_i^1$ into the attention module for cross-attention computations. This attention module predicts a supportiveness value w_j, which we use to adjust the contribution of $\boldsymbol{h}_j^2$ to the anchor's prediction $\boldsymbol{z}_i^1$ during training. The loss function of our BLNN can be written as:

$$\begin{aligned}\mathcal{L}_{BLNN} = &\underbrace{-\frac{1}{n}\sum_{i=1}^{n}\frac{\boldsymbol{z}_i^1 \cdot \boldsymbol{h}_i^2}{\| \boldsymbol{z}_i^1 \| \| \boldsymbol{h}_i^2 \|}}_{\text{Bootstrap Latents of Nodes}} \\ &\underbrace{-\frac{1}{n}\sum_{i=1}^{n}\sum_{j\in\mathcal{N}_i} w_j \frac{\boldsymbol{z}_i^1 \cdot \boldsymbol{h}_j^2}{\| \boldsymbol{z}_i^1 \| \| \boldsymbol{h}_j^2 \|}}_{\text{Bootstrap Latents of Neighbors}}.\end{aligned} \tag{3}$$

Attention Weighting. The attention module, which softly measure the positiveness of node-neighbor pairs, simply consists of a cross-attention operator, and a softmax activation. Formally, given the anchor's online representation $\boldsymbol{h}_i^1$ and its neighboring node's target representation $\boldsymbol{h}_j^2$, the supportiveness score can be computed as:

$$w_j = \mathrm{softmax}_j(e_{ij}) = \frac{\exp(e_{ij}/\tau)}{\sum_{k\in\mathcal{N}_i}\exp(e_{ik}/\tau)}, \tag{4}$$

where $e_{ij} = \boldsymbol{h}_i^1 \cdot \boldsymbol{h}_j^2 / \| \boldsymbol{h}_i^1 \| \| \boldsymbol{h}_j^2 \|$ is the cosine similarity between $\boldsymbol{h}_i^1$ and $\boldsymbol{h}_j^2$ and τ is a temperature parameter. This attention module assigns higher weights to ground-truth positive node-neighbor pairs than false positive node-neighbor pairs, thus mitigating class collision caused by aligning false node-neighbor pairs.

Comparison with BGRL. Our BLNN enhances BGRL by introducing potential positive node-neighbor pairs in the absence of ground-truth labels. It inherits BGRL's advantages, such as the negative-free property, which naturally address class collision caused by false negative pairs. Different from the original BGRL framework, which aligns only augmented views with the anchor node, the cross-attention design in BLNN enriches the diversity of positive nodes to support the anchor node in a soft and adaptive manner. This design empowers us to leverage more positive pairs, enhancing intra-class compactness. Additionally, the computations for supportiveness scores and node-neighbor alignment loss exhibit a time complexity linear with the number of edges $\mathcal{O}(|\mathcal{E}|)$. Given the sparsity of real-world graphs, i.e., $\mathcal{O}(|\mathcal{E}|) << \mathcal{O}(|\mathcal{V}|^2)$, such complexity increase compared to BGRL is acceptable and our model maintains lower time complexity than contrastive learning baselines [41,42,39,16].

Algorithm 1 Bootstrap Latents of Nodes and Neighbors

Input: $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$
Parameter: Temperature τ, BGRL-related hyperparameters
Output: The graph encoder f_θ

Initialize model parameters;
while not converge **do**
  Sample two augmentation functions $t^1, t^2 \sim \mathcal{T}$;
  Generate augmented views $(\boldsymbol{A}^1, \boldsymbol{X}^1), (\boldsymbol{A}^2, \boldsymbol{X}^2)$;
  Obtain online representations $\boldsymbol{H}^1 = f_\theta(\boldsymbol{A}^1, \boldsymbol{X}^1)$;
  Obtain target representations $\boldsymbol{H}^2 = f_\phi(\boldsymbol{A}^2, \boldsymbol{X}^2)$;
  Compute positiveness scores of node-neighbor pairs via Eq. (4);
  Predict the target representations $\boldsymbol{Z}^1 = p_\theta(\boldsymbol{H}^1)$;
  Calculate the objective function via Eq. (3);
  Update the parameters of f_θ, p_θ via SGD;
  Update the parameters of f_ϕ via an EMA of f_θ;
end while
return f_θ.

5 Experiments

In this section, we design the experiments to evaluate our proposed BLNN and answer the following research questions. **RQ1**: Does BLNN outperform existing baseline methods on node classification, node clustering, and node similarity search? **RQ2**: How does each component of BLNN benefit the performance? **RQ3**: Can the supportiveness score measure the positiveness of node-neighbor pairs? **RQ4**: Is BLNN sensitive to the hyperparameter τ? **RQ5**: How to intuitively understand BLNN can enhance intra-class compactness of learned representations?

5.1 Experiment Setup

Datasets. We adopt five publicly available real-world benchmark datasets, including one reference network WikiCS [22], two co-purchase networks Photo, Computer [26], and two co-authorship networks CS, Physics [26] to conduct the experiments throughout the paper. The statistics of the datasets are provided in Table 1. More details can be found in Appendix A.1.

Table 1: Dataset statistics. $\mathcal{H}$ is the fraction of intra-class node-neighbor pairs.

Dataset	#Nodes	#Edges	#Feats	#Classes	$\mathcal{H}$ (%)
WikiCS	11,701	431,726	300	10	65.47
Photo	7,650	238,163	745	8	82.72
Computer	13,752	491,722	767	10	77.72
CS	18,333	163,788	6,805	15	80.81
Physics	34,493	495,924	8,415	5	93.14

Baselines. We compare BLNN with a variety of baselines, including supervised methods MLP, GCN [14], and GAT [29]; contrastive methods DGI [30], MVGRL [9], GRACE [41], GCA [42], AF-GCL [31], COSTA [39], FastGCL [33], gCooL [16], ProGCL [35], and CGKS [38]; non-contrastive methods CCA-SSG [37], G-BT [1], AFGRL [15], GraphMAE [11], and BGRL [28]. All the baseline results are taken from previously published papers. And brief introductions of the baselines can be found in Appendix A.2.

Evaluation Protocol. We evaluate BLNN on three tasks, i.e., node classification, node clustering and node similarity search. We first train the model in an unsupervised manner. For node classification, we use the learned representations to train and test a simple logistic regression classifier with twenty 1:1:8 train/validation/test random splits (twenty public splits for WikiCS) [28]. We apply K-means to the learned representations, initializing the cluster numbers with fixed values. For node similarity search, we use pairwise cosine similarity to identify nearest node neighbors [15]. Evaluations are conducted at every 250 epochs, and we report the best results [28,15].

Metrics. Following AFGRL [15], we use accuracy for node classification, normalized mutual information (NMI) and homogeneity (Hom.) for node clustering. For node similarity search, we introduce S@k, which is average ratio among the k nearest neighbors sharing the same label as the query node. Formulas of these metrics can be found in Appendix A.4.

Implementation Details. Since our BLNN is derived from BGRL, we implement BLNN based on the official code[4] of BGRL. To ensure a fair comparison, all BGRL-related hyperparameters are the same as those specified in the original BGRL paper. We perform a grid-search on the introduced temperature hyperparameter τ. All experiments are conducted on a 32GB V100 GPU. Our implementation code is available at `https://github.com/Cloudy1225/BLNN`. More details can be found in Appendix A.5.

5.2 Experiment Results

Performance Analysis (RQ1). The experimental results of node classification are presented in Table 2, revealing that our BLNN outperforms both self-supervised and even supervised baselines. This superiority can be attributed to two primary factors: 1) The pioneering BGRL of BLNN can effectively learn discriminative node representations, achieving competitive performance. 2) BLNN introduces additional potential positive pairs, enhancing the intra-class compactness of representations learned by BGRL. Node clustering results are detailed in Table 3, demonstrating BLNN's superior performance across four datasets, except Physics. Notably, BLNN exhibits significant improvement over BGRL, especially on WikiCS, Computer and Physics, with an increase ranging from 5% to 8%. These enhancements underscore the effectiveness of incorporating positive node-neighbor pairs to generate more intra-class compact representations. Table 4 illustrates the node similarity search results, with BLNN demonstrating the best performance. This outcome aligns with expectations, as BLNN is designed to softly pull together representations of nodes and their neighbors, where neighboring nodes often share the same label in graphs.

[4] `https://github.com/nerdslab/bgrl`

Table 2: Node classification results measured by accuracy along with standard deviations. The baseline results are taken from previously published papers. '-' denotes the absence of the result in the original paper. The *Input* column illustrates the data used in the training stage, and $\boldsymbol{Y}$ denotes labels.

Method	Input	WikiCS	Photo	Computer	CS	Physics
MLP	$\boldsymbol{X}, \boldsymbol{Y}$	71.98±0.00	78.53±0.00	73.81±0.00	90.37±0.00	93.58±0.00
GCN	$\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{Y}$	77.19±0.12	92.42±0.22	86.51±0.54	93.03±0.31	95.65±0.16
GAT	$\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{Y}$	77.65±0.11	92.56±0.35	86.93±0.29	92.31±0.24	95.47±0.15
DGI	$\boldsymbol{A}, \boldsymbol{X}$	78.25±0.56	91.69±1.07	87.98±0.81	92.15±0.63	94.51±0.52
MVGRL	$\boldsymbol{A}, \boldsymbol{X}$	77.57±0.46	92.04±0.98	87.39±0.92	92.11±0.12	95.33±0.03
GRACE	$\boldsymbol{A}, \boldsymbol{X}$	78.64±0.33	92.46±0.18	88.29±0.11	92.17±0.04	95.26±0.22
GCA	$\boldsymbol{A}, \boldsymbol{X}$	78.35±0.05	92.53±0.16	87.85±0.31	93.10±0.01	95.68±0.05
AF-GCL	$\boldsymbol{A}, \boldsymbol{X}$	79.01±0.51	92.49±0.31	89.68±0.19	91.92±0.10	95.12±0.15
COSTA	$\boldsymbol{A}, \boldsymbol{X}$	79.12±0.02	92.56±0.45	88.32±0.03	92.94±0.10	95.60±0.02
FastGCL	$\boldsymbol{A}, \boldsymbol{X}$	79.20±0.07	92.91±0.07	89.35±0.09	92.71±0.07	95.53±0.02
gCooL	$\boldsymbol{A}, \boldsymbol{X}$	78.74±0.04	93.18±0.12	88.85±0.14	93.32±0.02	-
ProGCL	$\boldsymbol{A}, \boldsymbol{X}$	78.68±0.12	93.30±0.09	89.28±0.15	93.51±0.06	-
CGKS	$\boldsymbol{A}, \boldsymbol{X}$	79.20±0.10	92.40±0.10	88.50±0.20	93.00±0.20	-
CCA-SSG	$\boldsymbol{A}, \boldsymbol{X}$	79.08±0.53	93.14±0.14	88.74±0.28	93.32±0.22	95.38±0.06
G-BT	$\boldsymbol{A}, \boldsymbol{X}$	76.83±0.73	92.46±0.35	87.93±0.36	92.91±0.25	95.25±0.13
AFGRL	$\boldsymbol{A}, \boldsymbol{X}$	77.62±0.49	93.22±0.28	89.88±0.33	93.27±0.17	95.69±0.10
GraphMAE	$\boldsymbol{A}, \boldsymbol{X}$	79.54±0.58	92.98±0.35	89.88±0.10	93.08±0.17	95.40±0.06
BGRL	$\boldsymbol{A}, \boldsymbol{X}$	79.98±0.10	93.17±0.30	90.34±0.19	93.31±0.13	95.73±0.05
BLNN	$\boldsymbol{A}, \boldsymbol{X}$	**80.48±0.52**	**93.54±0.23**	**91.02±0.23**	**93.61±0.15**	**95.86±0.10**

Table 3: Performance on node clustering. The baseline results are taken from the published AFGRL paper.

Dataset	WikiCS		Photo		Computer		CS		Physics	
Metric	NMI	Hom.	NMI	Hom.	NMI	Hom.	NMI	Hom.	NMI	Hom.
GRACE	42.82	44.23	65.13	66.57	47.93	52.22	75.62	79.09	-	-
GCA	33.73	35.25	64.43	65.75	52.78	58.16	76.20	79.65	-	-
AFGRL	41.32	43.07	65.63	67.43	55.20	60.40	78.59	81.61	**72.89**	**73.54**
BGRL	39.69	41.56	68.41	70.04	53.64	58.69	77.32	80.41	55.68	60.18
BLNN	**47.17**	**49.11**	**71.05**	**72.18**	**58.79**	**64.33**	**78.97**	**82.08**	62.41	67.39

Ablation Studies (RQ2). To verify the benefit of each component of BLNN, we conduct ablation studies with different variants of BGRL: BGRL with raw nosiy node-neighbor pairs ($\text{BGRL}_{\text{noisy}}$), BGRL with clean node-neighbor pairs ($\text{BGRL}_{\text{clean}}$), and our proposed BLNN (BGRL with supportiveness-weighted node-neighbor pairs). Results are reported in Table 5. We can find that simply treating raw node-neighbor pairs as ground-truth positive pairs results in only marginal improvement or even performance degradation, as raw node-neighbor pairs include inter-class pairs, which would cause

Table 4: Performance on node similarity search. The baseline results are taken from the published AFGRL paper.

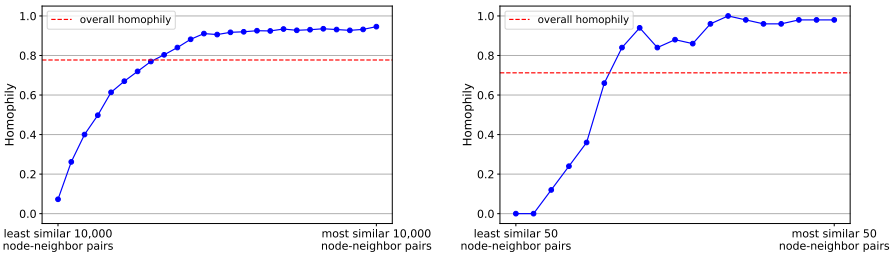
Dataset	WikiCS		Photo		Computer		CS		Physics	
Metric	S@5	S@10	S@5	S@10	S@5	S@10	S@5	S@10	S@5	S@10
GRACE	77.54	76.45	91.55	91.06	87.38	86.43	91.04	90.59	-	-
GCA	77.86	76.73	91.12	90.52	88.26	87.42	91.26	91.00	-	-
AFGRL	78.11	76.60	92.36	91.73	89.66	88.90	91.80	91.42	95.25	94.86
BGRL	77.39	76.17	92.45	91.95	89.47	88.55	91.12	90.86	95.04	94.64
BLNN	**80.27**	**79.04**	**92.61**	**91.96**	**89.91**	**89.12**	**91.90**	**91.59**	**95.39**	**95.01**

class collision. Our supportiveness weighting strategy, implemented through an attention module, effectively mitigates this class collision, yielding superior performance. However, there is still a gap between our BLNN and the ideal solution $\text{BGRL}_{\text{clean}}$, which necessitates the availability of all labels. These results further confirm our motivation described in Section 4.1.

Table 5: Ablation study on node classification.

Variant	WikiCS	Photo	Computer	CS	Physics
BGRL	79.98	93.17	90.34	93.31	95.73
BLNN	80.48	93.54	91.02	93.61	95.86
$\text{BGRL}_{\text{noisy}}$	80.05	93.33	90.44	93.27	95.59
$\text{BGRL}_{\text{clean}}$	81.51	93.66	91.31	93.92	95.98

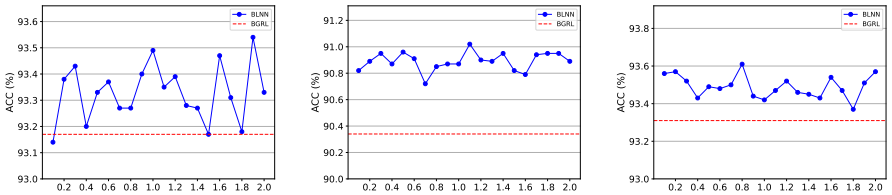
Case Study (RQ3). Our attention module is implemented based on cosine similarities of node-neighbor pairs and is expected to assign higher weights to true positive node-neighbor pairs than false positive pairs. Here, we conduct a twofold case study on Computer to verify that: 1) node-neighbor pairs with higher cosine similarity tend to share the same label; 2) our attention module indeed assigns higher weights to true positive node-neighbor pairs. We first sort all node-neighbor pairs based on the learned cosine similarity and then divide them into intervals of size $10,000$ to compute the homophily in each interval. As shown in Figure 3(a), the cosine similarity effectively estimates the probability of neighbor nodes being positive, with more similar node-neighbor pairs exhibiting larger homophily, which validates the efficacy of leveraging cosine similarity in our attention module. Moreover, we select an anchor node with 949 neighbors, sorting all anchor-neighbor pairs according to the supportiveness weights predicted by the attention module. We also partition them into intervals of size 50 to calculate homophily within each interval. As shown in Figure 3(b), our attention module generally assigns higher weights to true positive node-neighbor pairs compared to false positive pairs.



(a) all node-neighbor pairs (b) anchor-neighbor pairs

Fig. 3: Case study to verify the efficacy of our attention module.

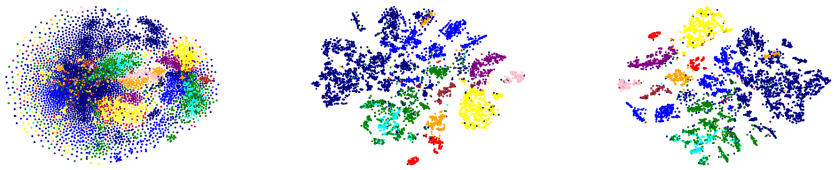
Hyperparameter Analysis (RQ4). We investigate the impact of the temperature τ in Eq. (4) on node classification by varying τ from 0.1 to 2.0 in increments of 0.1. Figure 4 presents the ACC scores on Photo, Computer and CS. It is observed that, our BLNN almost always achieves better performance than BGRL with respect to different τ. In general, BLNN exhibits robustness to the temperature τ. Analysis for BGRL-related hyperparameters can be found in the original BGRL paper [28].



(a) Photo (b) Computer (c) CS

Fig. 4: Visualization of the impact of τ on node classification.

Visualization and Compactness of Representations (RQ5). To gain a more intuitive insight into node representations, we provide the t-SNE [20] visualizations of the raw features and representations learned by BGRL and BLNN, along with intra-class compactness score on Computer. The intra-class compactness score is defined as the mean cosine similarity among all intra-class node pairs (the formula can be found in Appendix A.4). As shown in Figure 5, the representations learned by BLNN exhibit higher intra-class compactness, thus underscoring the effectiveness of mining positive node-neighbor pairs.



(a) Raw(0.4046) (b) BGRL(0.6733) (c) BLNN(0.7015)

Fig. 5: t-SNE visualization and intra-class compactness of node representations on Computer. '$(*)$' indicates the mean intra-class pair-wise cosine similarity.

6 Conclusion

In this paper, we introduce Bootstrap Latents of Nodes and Neighbors (BLNN). Our proposal is motivated by the empirical observation that introducing ground-truth positive node-neighbor pairs can yield significant improvements for BGRL. We thus expand the positive pair set with node-neighbor pairs and propose a cross-attention module to weight the contribution of each neighbor to loss computations. This module prioritizes higher weights for ground-truth positive node-neighbor pairs compared to false positive node-neighbor pairs, thereby alleviating class collision resulting from the alignment of false node-neighbor pairs. Extensive experiments demonstrate that our BLNN effectively improves the intra-class compactness of learned representations, establishing its state-of-the-art performance in three downstream tasks across five benchmark datasets.

Acknowledgments

This work is partially supported by the National Key Research and Development Program of China (2021YFB1715600), and the National Natural Science Foundation of China (62306137).

References

1. Bielak, P., Kajdanowicz, T., Chawla, N.V.: Graph barlow twins: A self-supervised representation learning framework for graphs. Knowledge-Based Systems **256**, 109631 (2022)
2. Chen, H., Zhao, Z., Li, Y., Zou, Y., Li, R., Zhang, R.: Csgcl: Community-strength-enhanced graph contrastive learning. In: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23. pp. 2059–2067 (2023)
3. Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., Zisserman, A.: With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9588–9597 (2021)
4. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)

5. GE, C., Wang, J., Tong, Z., Chen, S., Song, Y., Luo, P.: Soft neighbors are positive supporters in contrastive visual representation learning. In: The Eleventh International Conference on Learning Representations (2023)
6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: International Conference on Artificial Intelligence and Statistics (2010), https://api.semanticscholar.org/CorpusID:5575601
7. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Doersch, C., Pires, B.A., Guo, Z.D., Azar, M.G., Piot, B., Kavukcuoglu, K., Munos, R., Valko, M.: Bootstrap your own latent a new approach to self-supervised learning. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS'20 (2020)
8. Gugger, S., Howard, J.: Adamw and super-convergence is now the fastest way to train neural nets (2018), https://www.fast.ai/posts/2018-07-02-adam-weight-decay.html
9. Hassani, K., Khasahmadi, A.H.: Contrastive multi-view representation learning on graphs. In: Proceedings of the 37th International Conference on Machine Learning. ICML'20 (2020)
10. Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., Bengio, Y.: Learning deep representations by mutual information estimation and maximization. In: International Conference on Learning Representations (2019)
11. Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., Tang, J.: Graphmae: Self-supervised masked graph autoencoders. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 594–604 (2022)
12. Ji, W., Deng, Z., Nakada, R., Zou, J., Zhang, L.: The power of contrast for feature learning: A theoretical analysis. Journal of Machine Learning Research **24**(330), 1–78 (2023)
13. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. Advances in neural information processing systems **33**, 18661–18673 (2020)
14. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (2017)
15. Lee, N., Lee, J., Park, C.: Augmentation-free self-supervised learning on graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 7372–7380 (2022)
16. Li, B., Jing, B., Tong, H.: Graph communal contrastive learning. In: Proceedings of the ACM Web Conference 2022. p. 1203–1213. WWW'22 (2022)
17. Li, H., Cao, J., Zhu, J., Luo, Q., He, S., Wang, X.: Augmentation-free graph contrastive learning of invariant-discriminative representations. IEEE Transactions on Neural Networks and Learning Systems (2023)
18. Li, W.Z., Wang, C.D., Xiong, H., Lai, J.H.: Homogcl: Rethinking homophily in graph contrastive learning. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. p. 1341–1352. KDD '23, Association for Computing Machinery, New York, NY, USA (2023)
19. Liu, Y., Jin, M., Pan, S., Zhou, C., Zheng, Y., Xia, F., Yu, P.S.: Graph self-supervised learning: A survey. IEEE Trans. on Knowl. and Data Eng. **35**(6), 5879–5900 (2022)
20. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(86), 2579–2605 (2008)
21. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: Homophily in social networks. Review of Sociology **27**, 415–444 (2001)
22. Mernyei, P., Cangea, C.: Wiki-cs: A wikipedia-based benchmark for graph neural networks. ArXiv **abs/2007.02901** (2020)
23. Nowozin, S., Cseke, B., Tomioka, R.: f-gan: Training generative neural samplers using variational divergence minimization. Advances in neural information processing systems **29** (2016)

24. Peng, Z., Huang, W., Luo, M., Zheng, Q., Rong, Y., Xu, T., Huang, J.: Graph representation learning via graphical mutual information maximization. In: Proceedings of The Web Conference 2020. pp. 259–270 (2020)
25. Saunshi, N., Plevrakis, O., Arora, S., Khodak, M., Khandeparkar, H.: A theoretical analysis of contrastive unsupervised representation learning. In: International Conference on Machine Learning. pp. 5628–5637. PMLR (2019)
26. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. Relational Representation Learning Workshop, NeurIPS 2018 (2018)
27. Sun, W., Li, J., Chen, L., Wu, B., Bian, Y., Zheng, Z.: Rethinking and simplifying bootstrapped graph latents. In: Proceedings of the 17th ACM International Conference on Web Search and Data Mining. pp. 665–673 (2024)
28. Thakoor, S., Tallec, C., Azar, M.G., Azabou, M., Dyer, E.L., Munos, R., Veličković, P., Valko, M.: Large-scale representation learning on graphs via bootstrapping. In: International Conference on Learning Representations (2022)
29. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
30. Veličković, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. In: International Conference on Learning Representations (2019)
31. Wang, H., Zhang, J., Zhu, Q., Huang, W.: Augmentation-free graph contrastive learning with performance guarantee. arXiv preprint arXiv:2204.04874 (2022)
32. Wang, T., Isola, P.: Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In: Proceedings of the 37th International Conference on Machine Learning. vol. 119, pp. 9929–9939 (2020)
33. Wang, Y., Sun, W., Xu, K., Zhu, Z., Chen, L., Zheng, Z.: Fastgcl: Fast self-supervised learning on graphs via contrastive neighborhood aggregation (2022)
34. Wen, Y., Liu, W., Feng, Y., Raj, B., Singh, R., Weller, A., Black, M.J., Schölkopf, B.: Pairwise similarity learning is simple. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5308–5318 (2023)
35. Xia, J., Wu, L., Wang, G., Chen, J., Li, S.Z.: Progcl: Rethinking hard negative mining in graph contrastive learning. In: International Conference on Machine Learning (2021)
36. Yi, L., Liu, S., She, Q., McLeod, A.I., Wang, B.: On learning contrastive representations for learning with noisy labels. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16682–16691 (2022)
37. Zhang, H., Wu, Q., Yan, J., Wipf, D., Yu, P.S.: From canonical correlation analysis to self-supervised graph neural networks. In: Advances in Neural Information Processing Systems. vol. 34, pp. 76–89 (2021)
38. Zhang, Y., Chen, Y., Song, Z., King, I.: Contrastive cross-scale graph knowledge synergy. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 3422–3433 (2023)
39. Zhang, Y., Zhu, H., Song, Z., Koniusz, P., King, I.: Costa: Covariance-preserving feature augmentation for graph contrastive learning. Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2022)
40. Zheng, M., Wang, F., You, S., Qian, C., Zhang, C., Wang, X., Xu, C.: Weakly supervised contrastive learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10042–10051 (2021)
41. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Deep graph contrastive representation learning. ArXiv **abs/2006.04131** (2020)
42. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Graph contrastive learning with adaptive augmentation. In: Proceedings of the Web Conference 2021. p. 2069–2080 (2021)

A Experiments

A.1 Datasets

We evaluate our model on five representative datasets: WikiCS, Photo, Computer, CS and Physics. Their brief introductions are as follows:

- **WikiCS** [22] is a reference network constructed from Wikipedia. It comprises nodes corresponding to articles in the field of Computer Science, where edges are derived from hyperlinks. The dataset includes 10 distinct classes representing various branches within the field. The node features are computed as the average GloVe word embeddings of the respective articles.
- **Photo** and **Computer** [26] are networks constructed from Amazon's co-purchase relationships. Nodes represent goods, and edges indicate frequent co-purchases between goods. The node features are represented by bag-of-words encoding of product reviews, and class labels are assigned based on the respective product categories.
- **CS** and **Physics** [26] are co-authorship networks based on the Microsoft Academic Graph. Here, nodes are authors, that are connected by an edge if they co-authored a paper; node features represent paper keywords for each author's papers, and class labels indicate most active fields of study for each author.

For all datasets, we use the processed version provided by PyTorch Geometric Library [4]. All datasets are public available and do not have licenses.

A.2 Baselines

In this subsection, we give brief introductions of the baselines used in the paper which are not described in the main paper due to the space constraint.

- **GCN** [14] and **GAT** [29] are two popular supervised graph neural networks that exploit structural information, raw node features, and node labels from the training set.
- **DGI** [30] maximizes the mutual information between node representations and graph summary.
- **MVGRL** [9] maximizes the mutual information between the cross-view representations of nodes and graphs using graph diffusion.
- **GRACE** [41] performs graph augmentation on the input graph and considers node-node level contrast on both inter-view and intra-view levels.
- **GCA** [42] extends GRACE with adaptive augmentation that incorporates various priors for topological and semantic aspects of the graph.
- **AF-GCL** [31] is an augmentation-free graph contrastive learning method, wherein the self supervision signal is constructed based on the aggregated features.
- **COSTA** [39] proposes covariance-preserving feature augmentation to overcome the bias issue introduced by the topology graph augmentation in graph contrastive learning.

- **FastGCL** [33] contrasts weighted-aggregated and non-aggregated neighborhood information, rather than disturbing the graph topology and node attributes, to achieve faster training and convergence speeds.
- **gCooL** [16] extends GRACE by jointly learning the community partition and node representations in an end-to-end fashion, thereby directly leveraging the inherent community structure within a graph.
- **ProGCL** [35] extends GRACE by leveraging hard negtive samples via Expectation Maximization to fit the observed node-level similarity distribution. We adopt the ProGCL-weight version as no synthesis of new nodes is leveraged.
- **CGKS** [38] preserves diverse hierarchical information through graph coarsening and facilitates cross-scale information interactions among different coarse graphs.
- **CCA-SSG** [37] leverages classical Canonical Correlation Analysis to formulate a feature-level objective which can discard augmentation-variant information and prevent dimensional collapse.
- **G-BT** [1] utilizes a cross-correlation-based loss function instead of negative samples, which enjoys fewer hyperparameters and significantly reduced computation time.
- **AFGRL** [15] extends BGRL by creating an alternative graph view through the discovery of nodes sharing both local structural information and global semantics with the original graph.
- **GraphMAE** [11] is a masked graph auto-encoder that focuses on feature reconstruction with both a masking strategy and scaled cosine error.
- **BGRL** [28] adopts asymmetrical BYOL [7] structure to align node-itself pairs without relying on negative samples, thus avoiding a quadratic bottleneck and class collision.

A.3 Graph Augmentation

We employ two graph data augmentation strategies designed to enhance graph attributes and topology information, respectively. They are widely used in graph self-supervised learning [41,37,28].

Feature Masking. We randomly select a portion of the node features' dimensions and mask their elements with zeros. Formally, we first sample a random vector $\widetilde{\boldsymbol{m}} \in \{0,1\}^F$, where each dimension is drawn from a Bernoulli distribution with probability $1-p_m$, i.e., $\widetilde{m}_i \sim \mathcal{B}(1-p_m), \forall i$. Then, the masked node features $\widetilde{\boldsymbol{X}}$ are computed by $\|_{i=1}^{N} \boldsymbol{x}_i \odot \widetilde{\boldsymbol{m}}$, where $\odot$ denotes the Hadamard product and $\|$ represents the stack operation (i.e., concatenating a sequence of vectors along a new dimension).

Edge Dropping. In addition to feature masking, we stochastically drop a certain fraction of edges from the original graph. Formally, since we only remove existing edges, we first sample a random masking matrix $\widetilde{\boldsymbol{M}} \in \{0,1\}^{N\times N}$, with entries drawn from a Bernoulli distribution $\widetilde{\boldsymbol{M}}_{i,j} \sim \mathcal{B}(1-p_d)$ if $\boldsymbol{A}_{i,j}=1$ for the original graph, and $\widetilde{\boldsymbol{M}}_{i,j}=0$ otherwise. Here, p_d represents the probability of each edge being dropped. The corrupted adjacency matrix can then be computed as $\widetilde{\boldsymbol{A}} = \boldsymbol{A} \odot \widetilde{\boldsymbol{M}}$.

We jointly utilize these two methods to generate graph views. And the hyperparameter settings for graph augmentations are the same as those in BGRL [28].

A.4 Formulas of Metrics

We denote the ground-truth class labels as$\boldsymbol{Y} = [y_i]_{i=1}^n$ and the labels predicted by a classifier or clustering model as $\hat{\boldsymbol{Y}} = [\hat{y}_i]_{i=1}^n$.

Accuracy is determined as the proportion of correct predictions:

$$\mathrm{ACC} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}(y_i = \hat{y}_i), \tag{5}$$

where $\mathbb{I}$ denotes the indicator function.

Normalized Mutual Information (NMI) measures the mutual information between the true class labels and the cluster assignments, normalized by the entropy of the class labels and the entropy of the cluster assignments. It is defined as:

$$\mathrm{NMI} = \frac{2I(\boldsymbol{Y};\hat{\boldsymbol{Y}})}{H(\boldsymbol{Y}) + H(\hat{\boldsymbol{Y}})}, \tag{6}$$

where $I(\cdot)$ is the mutual information, and $H(\cdot)$ is the entropy.

Homogeneity measures the degree to which each cluster contains only members of a single class:

$$\mathrm{Homo.} = 1 - \frac{H(\boldsymbol{Y}|\hat{\boldsymbol{Y}})}{H(\boldsymbol{Y})}. \tag{7}$$

S@$\boldsymbol{k}$ denotes the percentage of the top k neighbors that belong to the same class. It is defined as:

$$\mathrm{S@}\,k = \frac{1}{nk}\sum_{i=1}^{n}\sum_{j\in\mathcal{N}_k(i)}\mathbb{I}(y_i = y_j), \tag{8}$$

where $\mathcal{N}_k(i)$ denotes the k nearest neighbor set of i.

Intra-class Compactness of node representations is defined as the mean cosine similarity among all intra-class node pairs:

$$\mathcal{C} = \frac{1}{K}\sum_{l=1}^{K}\frac{1}{|\boldsymbol{Y}=l|}\sum_{y_i=y_j=l}^{i\neq j}\cos(\boldsymbol{h}_i,\boldsymbol{h}_j), \tag{9}$$

where K is the number of unique classes, $|\boldsymbol{Y}=l|$ is the number of nodes belonging to class l, and $\cos(\boldsymbol{h}_i,\boldsymbol{h}_j)$ is the cosine similarity between node representations $\boldsymbol{h}_i,\boldsymbol{h}_j$.

A.5 Implementation Details

Since our BLNN is derived from BGRL, we implement BLNN based on the official code[5] of BGRL. To ensure a fair comparison, all BGRL-related hyperparameters are the same as those specified in the original BGRL paper. Specially, we use the AdamW optimizer [8] with weight decay set to 10^{-5}, and all models initialized using Glorot initialization [6]. The encoders f_θ, f_ϕ are implemented as GCN [14] and the predictor

[5] `https://github.com/nerdslab/bgrl`

p_θ used to predict the embedding of nodes across views is fixed to be a Multilayer Perceptron (MLP) with a single hidden layer. The decay rate t controlling the rate of updates of the target parameters ϕ is initialized to 0.99 and gradually increased to 1.0 over the course of training following a cosine schedule. We perform a grid-search on the introduced temperature hyperparameter τ. Other model architecture and training details can be found in the original BGRL paper [28]. All experiments are conducted on a 32GB V100 GPU. Our implementation code is available at `https://github.com/Cloudy1225/BLNN`.

Table 6: Comparison with HomoGCL on node classification. The BGRL* and HomoGCL results are taken from the original HomoGCL paper, with the BGRL* results reproduced by HomoGCL's authors.

	BGRL*	HomoGCL	BLNN	BGRL
Photo	92.80	93.53	93.54	90.17
Computer	88.23	90.01	91.02	90.34

A.6 Comparison with HomoGCL

We observed that a peer study [18], called HomoGCL, shares certain similarities with our method. HomoGCL leverages homophily by estimating the probability of neighbor nodes being positive via Gaussian Mixture Model. It then softly aligns the representations of node-neighbor pairs and directly aligns the cluster assignment vectors of node-neighbor pairs. We provide node classification results in Table 6. The BGRL* and HomoGCL results are taken from the original HomoGCL paper, with the BGRL* results reproduced by HomoGCL's authors. We can find that our BLNN exhibits nearly identical performance to HomoGCL on Photo and demonstrates a substantial improvement on Computer. Additionally, HomoGCL requires performing time-consuming K-means clustering on the entire set of node representations to estimate cluster assignments. Finally, we express our gratitude to the authors of HomoGCL for their outstanding contributions to the graph self-supervised learning community.