ProGCL: Rethinking Hard Negative Mining in Graph Contrastive Learning

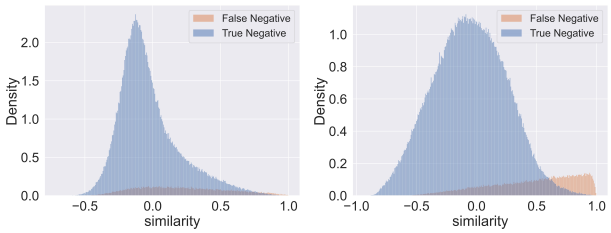
Jun Xia [1 2] **Lirong Wu** [1 2] **Ge Wang** [1 2] **Jintao Chen** [3] **Stan Z. Li** [1 2]

Abstract

Contrastive Learning (CL) has emerged as a dominant technique for unsupervised representation learning which embeds augmented versions of the anchor close to each other (positive samples) and pushes the embeddings of other samples (negatives) apart. As revealed in recent studies, CL can benefit from hard negatives (negatives that are most similar to the anchor). However, we observe limited benefits when we adopt existing hard negative mining techniques of other domains in Graph Contrastive Learning (GCL). We perform both experimental and theoretical analysis on this phenomenon and find it can be attributed to the message passing of Graph Neural Networks (GNNs). Unlike CL in other domains, most hard negatives are potentially false negatives (negatives that share the same class with the anchor) if they are selected merely according to the similarities between anchor and themselves, which will undesirably push away the samples of the same class. To remedy this deficiency, we propose an effective method, dubbed **ProGCL**, to estimate the probability of a negative being true one, which constitutes a more suitable measure for negatives' hardness together with similarity. Additionally, we devise two schemes (i.e., **ProGCL-weight** and **ProGCL-mix**) to boost the performance of GCL. Extensive experiments demonstrate that ProGCL brings notable and consistent improvements over base GCL methods and yields multiple state-of-the-art results on several unsupervised benchmarks or even exceeds the performance of supervised ones. Also, ProGCL is readily pluggable into various negatives-based GCL methods for performance improvement. We release the code at https://github.com/junxia97/ProGCL.

[1]Westlake University [2]Westlake Institute for Advanced Study [3]School of Computer Science, Zhejiang University. Correspondence to: Jun Xia <xiajun@westlake.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.



(a) CIFAR-10 (Image) (b) Coauthor-CS (Graph)

Figure 1. Similarity (cosine similarity between normalized embeddings of anchor and negatives) histograms of negatives. We adopt SimCLR (Chen et al., 2020) and GCA (Zhu et al., 2021c) for CIFAR-10 and Coauthor-CS respectively. False (or true) negatives denote the samples that are (or not) from the same class with the anchor. Unlike CL, most negatives with larger similarities to the anchor are false ones in GCL. More examples are in the appendix.

1. Introduction

Recently, Contrastive Learning (CL) has demonstrated unprecedented unsupervised performance in computer vision (He et al., 2020; Chen et al., 2020), natural language processing (Gao et al., 2021; Zheng et al., 2022) and graph representation learning (Velickovic et al., 2019; Xia et al., 2022b; Zhu et al., 2021c). As with metric learning (Kaya & Bilge, 2019), existing works both theoretically and practically validate that hard negatives contribute more to CL. For example, HCL (Robinson et al., 2021) develops a family of unsupervised sampling methods for selecting hard negatives and MoCHi (Kalantidis et al., 2020) mixes the hard negatives to synthesize more hard negative ones. However, we observe minor improvement or even significant performance drop when we adopt these negative mining techniques in GCL (the results can be seen in Table 5). Concurrent to our work, Zhu et al. (2021b) also observe that existing hard negative mining techniques that work well in other domains bring limited benefits to GCL. Unfortunately, they didn't provide any solution to tackle this issue.

To explain these phenomena, we first plot the negatives' distributions over similarity of various datasets in Figure 1. Note that we do not observe significant changes of negatives' distribution (keep unimodal as shown in Figure 1(a)) during the training process of SimCLR (Chen et al., 2020) on CIFAR-10 and other image datasets. However, for the

first stage of GCL, the negatives' distribution is bimodal for a long period as Figure 1(b) and then progressively transit to unimodal distribution as CL at the second stage. Similar phenomena for more datasets can be found in the appendix. The difference of negatives' distribution between CL and GCL can be attributed to the unique message passing of Graph Neural Networks, which we further discuss in section 3.2. These provide explanations for the poor performance of existing negative mining techniques in GCL. Specifically, they regard the negatives that are most similar to anchor points as hard ones across all the training process. However, as shown in Figure 1(b), most selected "hard" negatives in this way are false negatives for GCL indeed, which will undesirably push away the semantically similar samples and thus degrade the performance. The existence of false negatives is termed as *sampling bias* in DCL (Chuang et al., 2020). However, as reported in Table 5 and Zhu et al. (2021b), DCL brings performance drop for GCL because the sampling bias is severer in GCL. Now, we are naturally motivated to ask following questions: *Are there better alternatives to measure negatives' hardness considering the issue of false negatives in GCL? Can we devise more suitable methods to eliminate severer bias in GCL?*

To answer these questions, we argue that true and false negatives can be distinguished by fitting a two-component (true-false) beta mixture model (BMM) on the similarity. The posterior probability of a negative being true one under BMM can constitute a more suitable measure for negatives' hardness accompanied with similarity. With the novel measure, we devise two schemes (ProGCL-weight and ProGCL-mix) for further improvement of negatives-based GCL methods. To the best of our knowledge, our work makes one of the pioneering attempts to study hard negative mining in *node-level* GCL. We highlight the following contributions:

- We demonstrate the difference of negatives' distribution between GCL and CL and explain why existing hard negative mining techniques can not work well in GCL with both theoretical and experimental analysis.
- We propose to utilize BMM to estimate the probability of a negative being true one relative to a specific anchor. Combined with the similarity, we obtain a more suitable measure for negatives' hardness.
- We devise two schemes (i.e., ProGCL-weight and ProGCL-mix) that are more suitable for hard negative mining in GCL.
- ProGCL brings notable and consistent improvements over base GCL methods and yields multiple state-of-the-art results on several unsupervised benchmarks or even exceeds the performance of supervised ones. Also, it can boost various negatives-based GCL methods for further improvements.

2. Related Work

2.1. Graph Contrastive Learning (GCL)

Recently, GCL has gained popularity in unsupervised graph representation learning, which can get rid of the resource-intensive annotations (Xia et al., 2021; Tan et al., 2021; Xia et al., 2022a). Initially, DGI (Velickovic et al., 2019) and InfoGraph (Sun et al., 2020) are proposed to obtain expressive representations for graphs or nodes via maximizing the mutual information between graph-level representations and substructure-level representations of different granularity. Similarly, GMI (Peng et al., 2020) adopts two discriminators to directly measure mutual information between input and representations of both nodes and edges. Additionally, MVGRL (Hassani & Khasahmadi, 2020) proposes to learn both node-level and graph-level representation by performing node diffusion and contrasting node representation to augmented graph representation. Different from our work, GraphCL (You et al., 2020) and its variants (Suresh et al., 2021; Xu et al., 2021; You et al., 2021; 2022; Li et al., 2021) adopt SimCLR framework and design various augmentations for graph-level representation learning. For node-level representations, GRACE (Zhu et al., 2020) and its variants (Zhu et al., 2021c; Tong et al., 2021) maximize the agreement of node embeddings across two corrupted views of the graph. More recently, SimGRACE (Xia et al., 2022b), BGRL (Thakoor et al., 2021) and CCA-SSG (Zhang et al., 2021) try to simplify graph contrastive learning via discarding the negatives, parameterized mutual information estimator or even data augmentations. In this paper, we consider hard negatives mining to further boost GCL on *node-level representation learning*, which is still under-explored. For the limited space, we recommend readers refer to a recent review (Xia et al., 2022d) for more relevant literatures.

2.2. Hard Negative Mining in Contrastive Learning

For contrastive learning, HCL (Robinson et al., 2021) and MoCHi (Kalantidis et al., 2020) are proposed to emphasize or synthetize hard negatives. Additionally, Ring (Wu et al., 2021) introduces a family of mutual information estimators that sample negatives in a "ring" around each positive. However, as shown in Table 5, these techniques which emphasize hard negatives don't work well in GCL. For GCL, Zhao et al. (Zhao et al., 2021) utilize the clustering pseudo labels to alleviate false negative issue, which suffers from heavy computational overhead and will degrade the performance when confronted with multi-class datasets. CuCo (Chu et al., 2021) sorts the negatives from easy to hard with similarity for graph-level contrastive learning and proposes to automatically select and train negative samples with curriculum learning. Instead, we study node-level GCL with message passing among instances. Additionally, Zhu et al. (Zhu et al., 2022) propose a structure-enhanced negative mining scheme

which discovers hard negatives in each metapath-induced view in heterogeneous graph. However, it is not suitable for homogeneous graphs without metapath.

3. Methodology

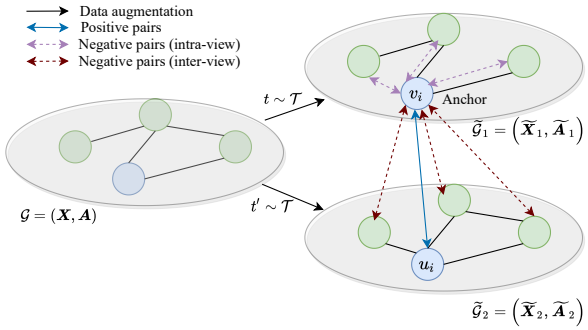


Figure 2. Schematic diagram of node-level GCL framework.

3.1. Preliminaries and Notations

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the graph, where $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}, \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denote the node set and edge set respectively. Additionally, $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ and $\boldsymbol{A} \in \{0,1\}^{N \times N}$ are the feature matrix and the adjacency matrix. $\boldsymbol{f}_i \in \mathbb{R}^F$ is the feature of v_i, and $\boldsymbol{A}_{ij} = 1$ iff $(v_i, v_j) \in \mathcal{E}$. Our objective is to learn a GNN encoder $f(\boldsymbol{X}, \boldsymbol{A}) \in \mathbb{R}^{N \times F'}$ to embed nodes in low dimensional space without label information. These low dimensional representations can be used in downstream tasks including node classification. As shown in Figure 2, we sample two augmentation functions $t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$, here $\mathcal{T}$ is the set of all augmentation functions. We can then obtain two views for $\mathcal{G}$, $\widetilde{\mathcal{G}}_1 = t(\mathcal{G})$ and $\widetilde{\mathcal{G}}_2 = t'(\mathcal{G})$. Given $\widetilde{\mathcal{G}}_1 = (\widetilde{\boldsymbol{X}}_1, \widetilde{\boldsymbol{A}}_1)$ and $\widetilde{\mathcal{G}}_2 = (\widetilde{\boldsymbol{X}}_2, \widetilde{\boldsymbol{A}}_2)$, we denote node embeddings in the two views as $\boldsymbol{U} = f(\widetilde{\boldsymbol{X}}_1, \widetilde{\boldsymbol{A}}_1)$ and $\boldsymbol{V} = f(\widetilde{\boldsymbol{X}}_2, \widetilde{\boldsymbol{A}}_2)$. For any node v_i, its embedding in one view $\boldsymbol{v}_i$ is regarded as the anchor. The embedding $\boldsymbol{u_i}$ in the other view is the positive sample and the embeddings of other nodes in both views are negatives. Similar to the InfoNCE (Oord et al., 2018), the training objective for each positive pair $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ is,

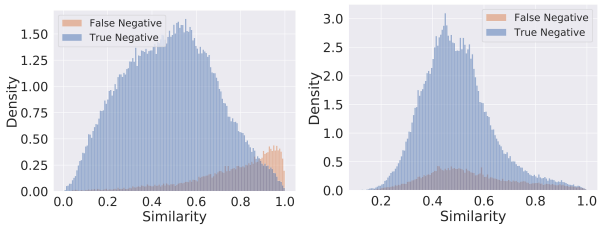
$$\ell(\boldsymbol{u}_i, \boldsymbol{v}_i) = \log \frac{e^{\theta(\boldsymbol{u}_i, \boldsymbol{v}_i)/\tau}}{\underbrace{e^{\theta(\boldsymbol{u}_i, \boldsymbol{v}_i)/\tau}}_{\text{positive pair}} + \underbrace{\sum_{k \neq i} e^{\theta(\boldsymbol{u}_i, \boldsymbol{v}_k)/\tau}}_{\text{inter-view negative pairs}} + \underbrace{\sum_{k \neq i} e^{\theta(\boldsymbol{u}_i, \boldsymbol{u}_k)/\tau}}_{\text{intra-view negative pairs}}}, \tag{1}$$

where the critic $\theta(\boldsymbol{u}, \boldsymbol{v}) = s(g(\boldsymbol{u}), g(\boldsymbol{v}))$. Here $s(\cdot, \cdot)$ is the cos similarity and $g(\cdot)$ is linear projection (two-layer perceptron model) to enhance the expression power of the critic function (Chen et al., 2020). With the symmetry of the two views, we can then define the overall loss as the average of all the positive pairs,

$$\mathcal{J} = -\frac{1}{2N} \sum_{i=1}^{N} \left[\ell(\boldsymbol{u_i}, \boldsymbol{v_i}) + \ell(\boldsymbol{v_i}, \boldsymbol{u_i})\right]. \tag{2}$$

Many GCL methods are built on this framework (Zhu et al., 2021b; Qiu et al., 2020; Zhu et al., 2021c; Jin et al., 2021), which motivates us to study hard negative mining on it.

3.2. Experimental and Theoretical Analysis



(a) GCN (b) MLP

Figure 3. Similarity (after Min-Max normalization) histograms of GCA (Zhu et al., 2021c) with GCN (with message passing) or MLP (without message passing) as encoder on Amazon-Photo.

In Figure 3, we study the message passing's role in GCL via replacing 2-layer GCN (Kipf & Welling, 2016a) that is composed of message passing and multi-layer perceptron (MLP) with a 2-layer MLP encoder only. As can be observed, the similarity histograms of GCL will be similar to that of CL without message passing, which verifies that message passing of GNN encoder is the key factor for the difference of negatives' distribution between CL and GCL. More specifically, for the first bimodal stage of GCL, the message passing in GCL enlarges the similarities between neighbour nodes which often share the same class. For the second unimodal stage, instance discrimination of GCL occupies a position of prominence and pushes away all the other samples regardless of their semantic class. Theoretically, for embeddings of any nodes pair in the graph $\mathcal{G}$, we can compare their distance before and after message passing and show that the distance will be decreased with the process, as formally induced in Theorem 3.1.

Theorem 3.1. *$\mathcal{G}$ is a non-bipartile and connected graph with N nodes $\mathcal{V} = \{v_1, \ldots, v_N\}$, and $\boldsymbol{X}_i^{(\tau)}$ is the embedding of node v_i after τ times message passing ($\boldsymbol{X}_i^{(0)} = \boldsymbol{f}_i$). For large enough τ, $||\boldsymbol{X}_i^{(\tau)} - \boldsymbol{X}_j^{(\tau)}||_2 \leq ||\boldsymbol{X}_i^{(0)} - \boldsymbol{X}_j^{(0)}||_2$.*

Proof. Given the message passing is,

$$\boldsymbol{X}^{(t+1)} = \hat{\boldsymbol{D}}^{-\frac{1}{2}} \hat{\boldsymbol{A}} \hat{\boldsymbol{D}}^{-\frac{1}{2}} \boldsymbol{X}^{(t)},$$

where $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ and $\hat{\boldsymbol{D}}_i = \sum_j \hat{\boldsymbol{A}}_{ij}$. We can then rewrite the message passing as,

$$\boldsymbol{X}^{(t+1)} = (\boldsymbol{I} - \boldsymbol{L}_{sym})\boldsymbol{X}^{(t)},$$

where $\boldsymbol{L}_{sym} = \hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{L}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}$, $\hat{\boldsymbol{L}} = \hat{\boldsymbol{D}} - \hat{\boldsymbol{A}}$. Let $(\lambda_1, \ldots, \lambda_N)$ and $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N)$ denote the eigenvalue and eigenvector of matrix $\boldsymbol{I} - \boldsymbol{L}_{sym}$, respectively. With the property of symmetric Laplacian matrix for non-bipartile and connected graph,

$$-1 < \lambda_1 < \lambda_2 < \cdots < \lambda_N = 1, \quad \boldsymbol{e}_N = \hat{\boldsymbol{D}}^{-\frac{1}{2}}[1, 1, \ldots, 1]^T,$$

we can rewrite $\boldsymbol{X}_i^{(\tau)} - \boldsymbol{X}_j^{(\tau)}$ as

$$\begin{aligned}\boldsymbol{X}_i^{(\tau)} - \boldsymbol{X}_j^{(\tau)} &= [(\boldsymbol{I} - \boldsymbol{L}_{sym})^\tau \boldsymbol{X}]_i - [(\boldsymbol{I} - \boldsymbol{L}_{sym})^\tau \boldsymbol{X}]_j \\ &= [\lambda_1^\tau(\boldsymbol{e}_1^{(i)} - \boldsymbol{e}_1^{(j)}), \ldots, \lambda_{n-1}^\tau(\boldsymbol{e}_{N-1}^{(i)} - \boldsymbol{e}_{N-1}^{(j)}), 0]\hat{\boldsymbol{X}}\end{aligned}$$

$\boldsymbol{e}_k^{(i)}$ is the i^{th} element of eigenvector $\boldsymbol{e}_k$, $\hat{\boldsymbol{X}}$ is the coordinate matrix of $\boldsymbol{X}$ in the space spanned by eigenvectors $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N)$. Thus,

$$||\boldsymbol{X}_i^{(\tau)} - \boldsymbol{X}_j^{(\tau)}||_2 = \sqrt{\sum_{m=1}^{N}[\sum_{k=1}^{N-1}\lambda_k^\tau(\boldsymbol{e}_k^{(i)} - \boldsymbol{e}_k^{(j)})\hat{\boldsymbol{X}}_{km}]^2}$$

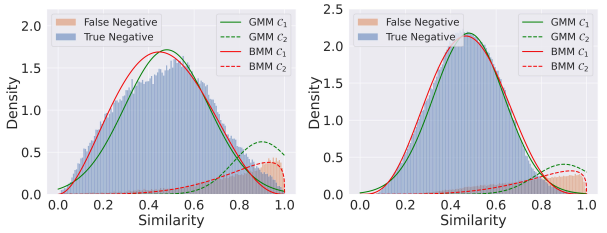
Because $1 < \lambda_1 < \lambda_2 < \cdots < \lambda_{N-1} < 1$, thus for a large τ, $||\boldsymbol{X}_i^{(\tau)} - \boldsymbol{X}_j^{(\tau)}||_2 \leq ||\boldsymbol{X}_i^{(0)} - \boldsymbol{X}_j^{(0)}||_2$. □

3.3. ProGCL

We aim to estimate the probability of a negative being true one. As can be observed in Figure 4, there is a significant difference between the false negatives and true negatives' distributions in GCL, allowing the two types of negatives can be distinguished from the similarity distribution. Here we propose to utilize mixture model to estimate the probability. Gaussian Mixture Model (GMM) is the most popular mixture model (Lindsay, 1995). However, in Figure 4, the distribution of false negatives is skew and thus symmetric Gaussian distribution can not fit this well. To circumvent this issue, we resort to beta distribution (Gupta & Nadarajah, 2004; Ji et al., 2005) which is flexible enough to model various distributions (symmetric, skewed, arched distributions and so on) over $[0, 1]$. As can be observed in Figure 4, Beta Mixture Model (BMM) can fit the empirical distribution better than GMM. Also, we compare the performance of ProGCL with BMM and GMM in Table 4 and find that BMM consistently outperforms GMM. The probability density function (pdf) of beta distribution is,

$$p(s \mid \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} s^{\alpha-1}(1-s)^{\beta-1}, \tag{3}$$

where $\alpha, \beta > 0$ are the parameters of beta distribution and $\Gamma(\cdot)$ is gamma function. The pdf of beta mixture model of



(a) Amazon-Photo (b) Coauthor-CS

Figure 4. Empirical distribution and estimated GMM and BMM on Amazon-Photo and Coauthor-CS datasets. The BMM fits the distributions better. Here $\mathcal{C}_1, \mathcal{C}_2$ denote the estimated distributions of two components respectively. Note that BMM is defined on the interval $[0, 1]$ while GMM is defined over $[-\infty, +\infty]$.

C components on s (Min-Max normalized cosine similarity between normalized embeddings of anchors and negatives) can be defined as:

$$p(s) = \sum_{c=1}^{C} \lambda_c p(s \mid \alpha_c, \beta_c), \tag{4}$$

where λ_c are the mixture coefficients. Here we can fit a two-component BMM (i.e., $C = 2$) to model the distribution of true and false negatives. We then utilize Expectation Maximization (EM) algorithm to fit BMM to the observed distribution. In E-step, we fix the parameters of BMM $(\lambda_c, \alpha_c, \beta_c)$ and update $p(c \mid s)$ with Bayes rule,

$$p(c \mid s) = \frac{\lambda_c p(s \mid \alpha_c, \beta_c)}{\sum_{j=1}^{C} \lambda_j p(s \mid \alpha_j, \beta_j)}. \tag{5}$$

In practice, fitting BMM with all the similarities will incur high computational cost. Instead, we can fit BMM well by only sampling M ($M \ll N^2$) similarities for simplification. Larger M bring limited benefits, which we will discuss in the appendix. We can then obtain the weighted average $\bar{s}_c$ and variance v_c^2 over M similarities,

$$\bar{s}_c = \frac{\sum_{i=1}^{M} p(c \mid s_i) s_i}{\sum_{i=1}^{M} p(c \mid s_i)}, \quad v_c^2 = \frac{\sum_{i=1}^{M} p(c \mid s_i)(s_i - \bar{s}_c)^2}{\sum_{i=1}^{M} p(c \mid s_i)}. \tag{6}$$

For M-step, the component model parameters α_c, β_c can be estimated using the method of moments in statistics,

$$\alpha_c = \bar{s}_c\left(\frac{\bar{s}_c(1-\bar{s}_c)}{v_c^2} - 1\right), \quad \beta_c = \frac{\alpha_c(1-\bar{s}_c)}{\bar{s}_c}, \tag{7}$$

and coefficients λ_c for mixing can be calculated as,

$$\lambda_c = \frac{1}{M}\sum_{i=1}^{M} p(c \mid s_i). \tag{8}$$

The above E and M-steps are iterated until convergence or the maximum of iterations I are reached. In our experiments, we set $I = 10$ and we study the influence of I in

the appendix. Finally, we can obtain the probability of a negative being true or negative one relative to the anchor with the their similarity s,

$$p\left(c \mid s\right)=\frac{\lambda_c p\left(s \mid \alpha_c, \beta_c\right)}{p\left(s\right)}. \tag{9}$$

Note that we can regard the fitted distribution with larger λ_c as true negatives' distribution because there are more true negatives than false ones relative to each anchor in multi-class datasets. Also, we can regard the fitted distribution with smaller mean as true negatives' distribution because the anchor and false negatives share the same class and they are more similar to each other in general. With the posterior probabilities, we devise two schemes to boost the performance of existing GCL as we elaborate below.

3.4. Scheme 1: ProGCL-weight

As revealed above, GCL suffers from severe sampling bias which will undermine the performance. To tackle this problem, we propose a novel measure for negatives' hardness considering the hardness and the probability of a negative being true one simultaneously,

$$w(i,k)=\frac{p\left(c_t \mid s_{ik}\right) s_{ik}}{\frac{1}{N-1}\sum_{j\neq i}\left[p\left(c_t \mid s_{ij}\right) s_{ij}\right]}, \tag{10}$$

where s_{ik} is the similarity between anchor $\boldsymbol{u}_i$ and its inter-view negative sample $\boldsymbol{v}_k$ and $p\left(c_t \mid s_{ij}\right)$ denotes the probability of $\boldsymbol{v}_j$ being true negative relative to anchor $\boldsymbol{u}_i$. Note $w(i,k)$ can be utilized to weight both inter-view $(\boldsymbol{u}_i, \boldsymbol{v}_k)$ and intra-view $(\boldsymbol{u}_i, \boldsymbol{u}_k)$ negative pair,

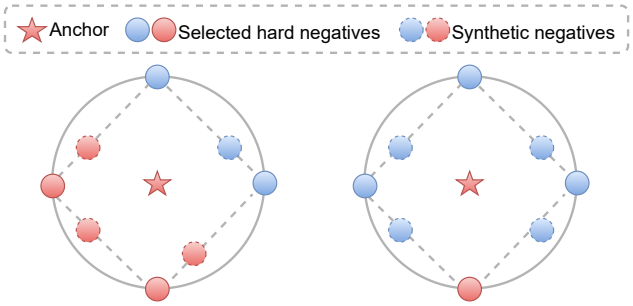
$$\ell_w\left(\boldsymbol{u}_i, \boldsymbol{v}_i\right)= \log \frac{e^{\frac{\theta\left(\boldsymbol{u}_i, \boldsymbol{v}_i\right)}{\tau}}}{\underbrace{e^{\frac{\theta\left(\boldsymbol{u}_i, \boldsymbol{v}_i\right)}{\tau}}}_{\text{positive pair}}+\underbrace{\sum_{k\neq i} w(i,k) e^{\frac{\theta\left(\boldsymbol{u}_i, \boldsymbol{v}_k\right)}{\tau}}}_{\text{inter-view negative pairs}}+\underbrace{\sum_{k\neq i} w(i,k) e^{\frac{\theta\left(\boldsymbol{u}_i, \boldsymbol{u}_k\right)}{\tau}}}_{\text{intra-view negative pairs}}}, \tag{11}$$

we can then define the new overall loss as the average of all the positive pairs,

$$\mathcal{J}_w=-\frac{1}{2N}\sum_{i=1}^{N}\left[\ell_w\left(\boldsymbol{u}_i, \boldsymbol{v}_i\right)+\ell_w\left(\boldsymbol{v}_i, \boldsymbol{u}_i\right)\right]. \tag{12}$$

3.5. Scheme 2: ProGCL-mix

Recently, MoCHi (Kalantidis et al., 2020) proposes to synthesize more negatives with "hard" negatives selected only with similarity. However, as analysed above, many synthesized hard negatives in GCL are positive samples indeed, which will undermine the performance. To remedy this deficiency, we propose ProGCL-mix which synthesizes more hard negatives considering the probability of a negative being true one. The comparison between MoCHi and ProGCL-mix can be seen in Figure 5. More specifically, for each



(a) MoCHi (b) ProGCL-mix

Figure 5. Comparison between MoCHi and ProGCL-mix. Gray dotted lines denote mixing and the classes are distinguished by the color of samples. ProGCL-mix synthesizes more true negatives.

anchor point $\boldsymbol{u}_i$, we synthesize m hard negatives by convex linear combinations of pairs of its "hardest" existing negatives. Here, "hardest" existing negatives refers to N' negatives that are selected with the measure in Eq. (10). Instead of mixing N' samples randomly, we mix them by emphasizing samples that are more likely to be true negative. Formally, for each anchor $\boldsymbol{u}_i$, a synthetic point $\tilde{\boldsymbol{u}}_k$ ($k \in [1, m]$) would be given by,

$$\tilde{\boldsymbol{u}}_k=\alpha_k \boldsymbol{v}_p+\left(1-\alpha_k\right)\boldsymbol{v}_q, \tag{13}$$

where $\boldsymbol{v}_p, \boldsymbol{v}_q$ are selected from N' "hardest" existing negatives measured by Eq (10) and α_k can be calculated as,

$$\alpha_k=\frac{p\left(c_t \mid s_{ip}\right)}{p\left(c_t \mid s_{ip}\right)+p\left(c_t \mid s_{iq}\right)}, \tag{14}$$

we can then define the training objective for each positive pair $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ with the synthetic negatives,

$$\ell_m\left(\boldsymbol{u}_i, \boldsymbol{v}_i\right)= \log \frac{e^{\frac{\theta\left(\boldsymbol{u}_i, \boldsymbol{v}_i\right)}{\tau}}}{\underbrace{e^{\frac{\theta\left(\boldsymbol{u}_i, \boldsymbol{v}_i\right)}{\tau}}}_{\text{positive pair}}+\underbrace{\sum_{k\neq i} e^{\frac{\theta\left(\boldsymbol{u}_i, \boldsymbol{v}_k\right)}{\tau}}}_{\text{inter-view negative pairs}}+\underbrace{\sum_{k\neq i} e^{\frac{\theta\left(\boldsymbol{u}_i, \boldsymbol{u}_k\right)}{\tau}}}_{\text{intra-view negative pairs}}+\underbrace{\sum_{k=1}^{m} e^{\frac{\theta\left(\boldsymbol{u}_i, \tilde{\boldsymbol{u}}_k\right)}{\tau}}}_{\text{synthetic negative pairs}}}. \tag{15}$$

Note that we synthesize new samples only with inter-view hard negatives. Finally, we can define the new overall loss,

$$\mathcal{J}_m=-\frac{1}{2N}\sum_{i=1}^{N}\left[\ell_m\left(\boldsymbol{u}_i, \boldsymbol{v}_i\right)+\ell_m\left(\boldsymbol{v}_i, \boldsymbol{u}_i\right)\right]. \tag{16}$$

3.6. Time Complexity Analysis

It is noteworthy that the estimation of the posterior probabilities introduces light computational overhead over the base model. Firstly, we only have to fit BMM once during the training process instead of once per epoch. Secondly, we can fit BMM well with M ($M \ll N^2$) similarities and the time complexity of EM algorithm for fitting in ProGCL

is $\mathcal{O}(IM)$. I is the the maximum of iterations. Thirdly, we only have to fit BMM with similarities from single view because both inter-view and intra-view include all negative pairs. In our experiments, we only utilize similarities from inter-view to fit BMM. The training algorithms of both ProGCL-weight and ProGCL-mix for transductive tasks are summarized in Algorithm 1. The algorithm for inductive tasks can be found in the appendix.

Algorithm 1 ProGCL-weight & -mix (Transductive)

Input: $\mathcal{T}, \mathcal{G}, f, g, N$, normalized cosine similarity s, epoch for fitting BMM E, $mode$ ('weight' or 'mix').
for $epoch = 0, 1, 2, ...$ **do**
  Draw two augmentation functions $t \sim \mathcal{T}, t' \sim \mathcal{T}$
  $\widetilde{\mathcal{G}}_1 = t(\mathcal{G}), \widetilde{\mathcal{G}}_2 = t'(\mathcal{G})$;
  $\mathcal{U} = f(\widetilde{\mathcal{G}}_1), \mathcal{V} = f(\widetilde{\mathcal{G}}_2)$;
  for $\boldsymbol{u}_i \in \mathcal{U}$ and $\boldsymbol{v}_i \in \mathcal{V}$ **do**
    $s_{ij} = s(g(\boldsymbol{u}_i), g(\boldsymbol{v}_i))$
    if $epoch = E$ **then**
      Compute $p\left(c_t \mid s_{ij}\right)$ with Eq. (4) to Eq. (9).
    end if
  end for
  if $epoch \geq E$ **then**
    if $mode$ = 'weight' **then**
      Compute $\mathcal{J}_w$ with Eq. (10) to Eq. (12).
      Update the parameters of f, g with $\mathcal{J}_w$;
    end if
    if $mode$ = 'mix' **then**
      Compute $\mathcal{J}_m$ with Eq. (13) to Eq. (16).
      Update the parameters of f, g with $\mathcal{J}_m$.
    end if
  else
    Compute $\mathcal{J}$ with Eq. (1) to Eq. (2).
    Update the parameters of f, g with $\mathcal{J}$.
  end if
end for
Output: f, g.

4. Experiments

4.1. Experimental Setup & Baselines

Following previous work (Velickovic et al., 2019), we train the model in an unsupervised manner. The resulting embeddings are utilized to train and test a simple ℓ_2-regularized logistic classifier. We train the classifier for 20 runs. Additionally, we adopt GRACE as base model and measure performance using micro-averaged F1-score on inductive tasks. For transductive tasks, we adopt GCA as base model and report the test accuracy.

Transductive learning. We adopt a two-layer GCN (Kipf & Welling, 2016a) as the encoder for transductive learning following previous works (Velickovic et al., 2019; Zhu et al., 2020). We consider our ProGCL with multiple baselines including DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016). Additionally, we also consider recent methods including Graph AutoEncoders (GAE, VGAE) (Kipf & Welling, 2016b), DGI, GMI, MVGRL, MERIT, BGRL and GCA as introduced in the related work. We report the best performance of three variants of GCA. We also compare ProGCL with supervised counterparts including GCN (Kipf & Welling, 2016a) and Graph Attention Networks (GAT) (Veličković et al., 2017).

Inductive learning on large graphs. Considering the large scale of some graph datasets, we adopt a three-layer GraphSAGE-GCN with residual connections as the encoder following DGI. We adopt the sub-sampling strategy in GraphSAGE where we first select a mini-batch of nodes and then a subgraph centered around each selected node is obtained by sampling node neighbors with replacement. More specifically, we sample 10, 10 and 25 neighbors at the first, second and third level respectively as DGI. The batchsize of our experiments is 256. Also, we estimate the posterior with pairwise similarities among each mini-batch instead of total training set, which we elaborate on in the appendix. We set traditional methods DeepWalk and deep learning based methods unsupervised GraphSAGE, DGI, a recent block-contrastive method COLES (Zhu et al., 2021a) and GMI as baselines. To compare ProGCL with supervised counterparts, we report the performance of two supervised methods FastGCN (Chen et al., 2018) and GraphSAGE. More details can be seen in the appendix.

4.2. Datasets

We conduct experiments on seven widely-used datasets including Amazon-Photo, Amazon-Computers, Wiki-CS, Coauthor-CS, Reddit, Flickr and ogbn-arXiv. To keep fair, *for transductive tasks*, we split Amazon-Photo, Amazon-Computers, Wiki-CS and Coauthor-CS for the training, validation and testing following (Zhu et al., 2021c). *For inductive task*, we split Reddit and Flickr following (Velickovic et al., 2019; Zeng et al., 2019). The experimental setting of ogbn-arXiv is the same as BGRL (Thakoor et al., 2021). More information of the datasets is in the appendix.

4.3. Comparison with State-of-the-art Results

For transductive classification, as can be observed in Table 1, ProGCL consistently performs better than previous unsupervised baselines or even the supervised baselines, which validates the superiority of our ProGCL. We provide more observations as following. Firstly, traditional methods node2vec and DeepWalk only using adjacency matrix outperform the simple logistic regression classifier that only uses raw features ("Raw features") on Amazon datasets. However, the latter can perform better on Coauthor-CS

Table 1. Summary of the accuracies (± std) on transductive node classification. The 'Available Data' refers to data we can obtain for training, where $\boldsymbol{X}$, $\boldsymbol{A}$ and $\boldsymbol{Y}$ denotes feature matrix, adjacency matrix and label matrix respectively. '**OOM**': out of memory on a 32GB GPU. We highlight the performance of ProGCL with gray background. The highest performance of unsupervised models is highlighted in boldface; the highest performance of supervised models is underlined. The baselines marked with '*' are reproduced with the same experimental settings (20 random dataset splits and model initializations). The other results are taken from previously published reports.

Method	Available Data	Amazon-Photo	Amazon-Computers	Coauthor-CS	Wiki-CS
Raw features	$\boldsymbol{X}$	78.53 ± 0.00	73.81 ± 0.00	90.37 ± 0.00	71.98 ± 0.00
node2vec	$\boldsymbol{A}$	89.67 ± 0.12	84.39 ± 0.08	85.08 ± 0.03	71.79 ± 0.05
DeepWalk	$\boldsymbol{A}$	89.44 ± 0.11	85.68 ± 0.06	84.61 ± 0.22	74.35 ± 0.06
DeepWalk + features	$\boldsymbol{X}, \boldsymbol{A}$	90.05 ± 0.08	86.28 ± 0.07	87.70 ± 0.04	77.21 ± 0.03
GAE	$\boldsymbol{X}, \boldsymbol{A}$	91.62 ± 0.13	85.27 ± 0.19	90.01 ± 0.17	70.15 ± 0.01
VGAE	$\boldsymbol{X}, \boldsymbol{A}$	92.20 ± 0.11	86.37 ± 0.21	92.11 ± 0.09	75.35 ± 0.14
DGI	$\boldsymbol{X}, \boldsymbol{A}$	91.61 ± 0.22	83.95 ± 0.47	92.15 ± 0.63	75.35 ± 0.14
GMI	$\boldsymbol{X}, \boldsymbol{A}$	90.68 ± 0.17	82.21 ± 0.31	**OOM**	74.85 ± 0.08
MVGRL*	$\boldsymbol{X}, \boldsymbol{A}$	92.08 ± 0.01	87.45 ± 0.21	92.18 ± 0.15	77.43 ± 0.17
BGRL*	$\boldsymbol{X}, \boldsymbol{A}$	92.95 ± 0.07	87.89 ± 0.10	92.72 ± 0.03	78.41 ± 0.09
MERIT*	$\boldsymbol{X}, \boldsymbol{A}$	92.53 ± 0.15	88.01 ± 0.12	92.51 ± 0.14	78.35 ± 0.05
GCA*	$\boldsymbol{X}, \boldsymbol{A}$	92.55 ± 0.03	87.82 ± 0.11	92.40 ± 0.07	78.26 ± 0.06
ProGCL-weight	$\boldsymbol{X}, \boldsymbol{A}$	93.30 ± 0.09	89.28 ± 0.15	93.51 ± 0.06	**78.68 ± 0.12**
ProGCL-mix	$\boldsymbol{X}, \boldsymbol{A}$	**93.64 ± 0.13**	**89.55 ± 0.16**	**93.67 ± 0.12**	78.45 ± 0.04
Supervised GCN	$\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y}$	92.42 ± 0.22	86.51 ± 0.54	93.03 ± 0.31	77.19 ± 0.12
Supervised GAT	$\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y}$	92.56 ± 0.35	86.93 ± 0.29	92.31 ± 0.24	77.65 ± 0.11

Table 2. Summary of the micro-averaged F_1 scores (± std) on inductive node classification.

Method	Available Data	Flickr	Reddit
Raw features	$\boldsymbol{X}$	20.3	58.5
DeepWalk	$\boldsymbol{A}$	27.9	32.4
GraphSAGE	$\boldsymbol{X}, \boldsymbol{A}$	36.5	90.8
DGI	$\boldsymbol{X}, \boldsymbol{A}$	42.9±0.1	94.0±0.1
GMI	$\boldsymbol{X}, \boldsymbol{A}$	44.5±0.2	94.8±0.0
COLES-S^{2}GC	$\boldsymbol{X}, \boldsymbol{A}$	46.8±0.5	95.2±0.3
GRACE	$\boldsymbol{X}, \boldsymbol{A}$	48.0±0.1	94.2±0.0
ProGCL-weight	$\boldsymbol{X}, \boldsymbol{A}$	49.2±0.6	95.1±0.2
ProGCL-mix	$\boldsymbol{X}, \boldsymbol{A}$	**50.0±0.3**	**95.6±0.1**
Supervised FastGCN	$\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y}$	48.1±0.5	89.5±1.2
Supervised GraphSAGE	$\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y}$	50.1±1.3	92.1±1.1

and Wiki-CS. Combing the both ("DeepWalk + features") can bring significant improvements. Compared with GCA, our ProGCL emphasize the hard negatives or remove sampling bias, which lifts the representation quality. Secondly, ProGCL-mix performs better than ProGCL-weight in general. For inductive tasks, ProGCL also achieves competitive performance over other baselines as shown in Table 2.

Table 3. Performance on the ogbn-arXiv measured in accuracy along with standard deviations. The results of baselines are taken from published reports. '−' means that the results are unavailable.

	Validation	Test
MLP	57.65 ± 0.12	55.50 ± 0.23
node2vec	71.29 ± 0.13	70.07 ± 0.13
Random-Init	69.90 ± 0.11	68.94 ± 0.15
DGI	71.26 ± 0.11	70.34 ± 0.16
GRACE-Subsampling	72.61 ± 0.15	71.51 ± 0.11
BGRL	72.53 ± 0.09	71.64 ± 0.12
COLES-S^{2}GC	–	72.48 ± 0.25
ProGCL-weight	72.45 ± 0.21	72.18 ± 0.09
ProGCL-mix	**72.82 ± 0.08**	**72.56 ± 0.20**
Supervised GCN	73.00 ± 0.17	71.74 ± 0.29

4.4. Results on Large-Scale OGB Dataset

We conduct experiments on another large graph datasets ogbn-arxiv from OGB benchmark (Hu et al., 2020). We adopt GRACE with sub-sampling as our base model and a 3-layer GCN as the encoder following Hu et al. (2020). As can observed in Table 3, ProGCL outperforms other unsupervised baselines, which verifies that ProGCL can achieve good tradeoff between performance and complexity. We report the results on both validation and test sets, as is convention for this task since the dataset is split based on a chronological ordering.

4.5. Improving Various GCL Methods

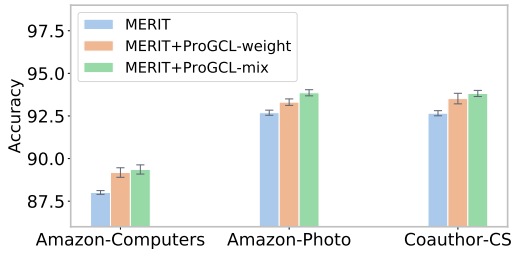
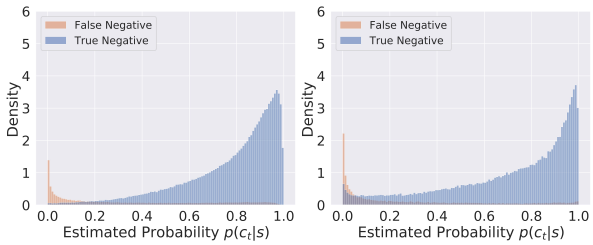


Figure 6. The performance of ProGCL for another graph contrastive learning method MERIT (Jin et al., 2021).

In addition to GCA and GRACE, we also evaluate the performance of ProGCL on another GCL method MERIT. The results shown in Figure 6 demonstrate that ProGCL brings consistent improvements over the base method, which verifies that our ProGCL is readily pluggable into various negatives-based GCL methods to improve their performance.

4.6. Why ProGCL Can Alleviate the Bias?



(a) Coauthor-CS (b) Amazon-Photo

Figure 7. The histograms of estimated probability.

To verify that our ProGCL can alleviate the sampling bias, we plot estimated probability histograms of negatives in Figure 7. Compared with similarity, the estimated probability serves as a more discriminative measure to distinguish true and false negatives, which can help us select hard and true negatives together with similarity as in Eq. (10).

4.7. Ablation Study

In this section, we replace or remove various parts of ProGCL to study the impact of each component.

Table 4. Comparison between BMM and GMM.

Datasets	Amazon-Photo		Amazon-Computers		Coauthor-CS	
Scheme	weight	mix	weight	mix	weight	mix
GMM	92.71	92.83	88.35	89.29	92.79	92.79
BMM	**93.30**	**93.64**	**89.28**	**89.55**	**93.51**	**93.67**

BMM vs. GMM. We replace BMM in our ProGCL with GMM and report the performance in Table 4. BMM consistently outperforms GMM in both weight and mix schemes. The reason is that BMM can fit the negatives distribution better than GMM as shown in Figure 4.

Table 5. Comparison between ProGCL and other hard nagative mining methods. "↑" and "↓" refer to performance improvement and drop compared with GCA respectively.

Methods/Datasets	Amazon-Photo	Amazon-Computers	Coauthor-CS
GCA	92.55	87.82	92.40
+DCL	91.02 (↓ 1.53)	86.58 (↓ 1.24)	92.36 (↓ 0.04)
+HCL	91.48 (↓ 1.07)	87.21 (↓ 0.61)	93.06 (↑ 0.66)
+MoCHi	92.36 (↓ 0.19)	87.68 (↑ 0.14)	92.58 (↑ 0.18)
+Ring	91.33 (↓ 1.22)	84.18 (↓ 3.64)	92.48 (↓ 0.08)
+ProGCL-mix	**93.64** (↑ 1.09)	**89.55** (↑ 1.73)	**93.67** (↑ 1.27)

ProGCL vs. Negative mining techniques. To study whether ProGCL can better utilize hard negatives and remove sampling bias in GCL, we equip GCA with DCL (Chuang et al., 2020), HCL (Robinson et al., 2021), Ring (Wu et al., 2021) and MoCHi (Kalantidis et al., 2020) which have achieved immense success in CL. As shown in Table 5, these techniques bring limited benefits over GCA. Instead, ProGCL introduces consistent and significant improvements over GCA, which further validates that our ProGCL is more suitable for GCL.

Table 6. Comparison between $p(c_t|s)$ and other alternatives.

Datasets	Amazon-Photo		Amazon-Computers		Coauthor-CS	
Alternatives	weight	mix	weight	mix	weight	mix
p_r	92.03	92.58	87.35	88.85	92.06	92.71
p_t	92.49	92.76	88.64	89.15	92.45	93.22
$p(c_t\|s)$	**93.30**	**93.64**	**89.28**	**89.55**	**93.51**	**93.67**

Performance attributable to ProGCL. We substitute the estimated probability $p(c_t|s)$ of ProGCL with random probability p_r and tuned p_t (1 - normalized similarity). As shown in Table 6, p_t can bring minor improvement over random p_r while the estimated posterior $p(c_t|s)$ by BMM is the best among the three alternatives.

4.8. Hyperparameters Sensitivity Analysis

Here we study the number of most hard negatives N' and synthetic negatives m of ProGCL-mix. The results shown in Figure 8 illustrate that more samples mixing bring consistent performance gains in general. However, oversized N' demonstrated no significant advantages in both accuracy and efficiency.

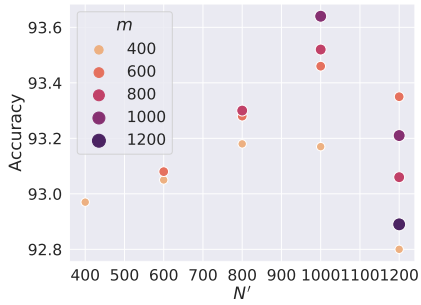


Figure 8. Accuracy when varying N' (x-axis) and m on Amazon-Photo. We study more hyperparameters in the appendix.

5. Conclusions

In this paper, we explain why existing hard negative mining methods can not work well in GCL and contrapuntally introduce BMM to estimate the probability of a negative being true one. Also, we devise two schemes to further boost GCL. Interesting directions of future work include (1) applying GCL to more real-world tasks including social analysis and drug discovery (Sun et al., 2021; Xia et al., 2022c); (2) exploring the theoretical explanations for the immense success of contrastive learning.

6. Acknowledgements

This work is supported in part by the Science and Technology Innovation 2030 - Major Project (No. 2021ZD0150100) and National Natural Science Foundation of China (No. U21A20427).

References

Chen, J., Ma, T., and Xiao, C. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

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. PMLR, 2020.

Chu, G., Wang, X., Shi, C., and Jiang, X. Cuco: Graph representation with curriculum contrastive learning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2021.

Chuang, C.-Y., Robinson, J., Yen-Chen, L., Torralba, A., and Jegelka, S. Debiased contrastive learning. *arXiv preprint arXiv:2007.00224*, 2020.

Gao, T., Yao, X., and Chen, D. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

Gupta, A. K. and Nadarajah, S. *Handbook of beta distribution and its applications*. CRC press, 2004.

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

Hassani, K. and Khasahmadi, A. H. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pp. 4116–4126. PMLR, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

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. *arXiv preprint arXiv:2005.00687*, 2020.

Ji, Y., Wu, C., Liu, P., Wang, J., and Coombes, K. R. Applications of beta-mixture models in bioinformatics. *Bioinformatics*, 21(9):2118–2122, 2005.

Jin, M., Zheng, Y., Li, Y.-F., Gong, C., Zhou, C., and Pan, S. Multi-scale contrastive siamese networks for self-supervised graph representation learning. In *The 30th International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

Kalantidis, Y., Sariyildiz, M. B., Pion, N., Weinzaepfel, P., and Larlus, D. Hard negative mixing for contrastive learning. In *Neural Information Processing Systems (NeurIPS)*, 2020.

Kaya, M. and Bilge, H. Ş. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

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

Li, H., Wang, X., Zhang, Z., Yuan, Z., Li, H., and Zhu, W. Disentangled contrastive learning on graphs. *Advances in Neural Information Processing Systems*, 34, 2021.

Lindsay, B. G. Mixture models: theory, geometry and applications. In *NSF-CBMS regional conference series in probability and statistics*, pp. i–163. JSTOR, 1995.

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

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Peng, Z., Huang, W., Luo, M., Zheng, Q., Rong, Y., Xu, T., and Huang, J. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, pp. 259–270, 2020.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., and Tang, J. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1150–1160, 2020.

Robinson, J. D., Chuang, C.-Y., Sra, S., and Jegelka, S. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=CR1XOQ0UTh-.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Sinha, A., Shen, Z., Song, Y., Ma, H., Eide, D., Hsu, B.-J., and Wang, K. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pp. 243–246, 2015.

Sun, F.-Y., Hoffman, J., Verma, V., and Tang, J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1lfF2NYvH.

Sun, M., Xing, J., Wang, H., Chen, B., and Zhou, J. Mocl: data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3585–3594, 2021.

Suresh, S., Li, P., Hao, C., and Neville, J. Adversarial graph augmentation to improve graph contrastive learning. *NeurIPS*, 2021.

Tan, C., Xia, J., Wu, L., and Li, S. Z. Co-learning: Learning from noisy labels with self-supervision. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 1405–1413, 2021.

Thakoor, S., Tallec, C., Azar, M. G., Munos, R., Veličković, P., and Valko, M. Bootstrapped representation learning on graphs. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*, 2021. URL https://openreview.net/forum?id=QrzVRAA49Ud.

Tong, Z., Liang, Y., Ding, H., Dai, Y., Li, X., and Wang, C. Directed graph contrastive learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax. In *ICLR (Poster)*, 2019.

Wu, M., Mosse, M., Zhuang, C., Yamins, D., and Goodman, N. Conditional negative sampling for contrastive learning of visual representations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=v8b3e5jN66j.

Xia, J., Lin, H., Xu, Y., Wu, L., Gao, Z., Li, S., and Li, S. Z. Towards robust graph neural networks against label noise, 2021. URL https://openreview.net/forum?id=H38f_9b90BO.

Xia, J., Tan, C., Wu, L., Xu, Y., and Li, S. Z. Ot cleaner: Label correction as optimal transport. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3953–3957, 2022a. doi: 10.1109/ICASSP43922.2022.9747279.

Xia, J., Wu, L., Chen, J., Hu, B., and Li, S. Z. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pp. 1070–1079, New York, NY, USA, 2022b. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3512156. URL https://doi.org/10.1145/3485447.3512156.

Xia, J., Zheng, J., Tan, C., Wang, G., and Li, S. Z. Towards effective and generalizable fine-tuning for pre-trained molecular graph models. *bioRxiv*, 2022c.

Xia, J., Zhu, Y., Du, Y., and Li, S. Z. A survey of pretraining on graphs: Taxonomy, methods, and applications. *arXiv preprint arXiv:2202.07893*, 2022d.

Xu, D., Cheng, W., Luo, D., Chen, H., and Zhang, X. InfoGCL: Information-aware graph contrastive learning. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=519VBzfEaKW.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. Graph contrastive learning with augmentations. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 5812–5823. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/3fe230348e9a12c13120749e3f9fa4cd-Paper.pdf.

You, Y., Chen, T., Shen, Y., and Wang, Z. Graph contrastive learning automated. *arXiv preprint arXiv:2106.07594*, 2021.

You, Y., Chen, T., Wang, Z., and Shen, Y. Bringing your own view: Graph contrastive learning without prefabricated data augmentations, 2022.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Zhang, H., Wu, Q., Yan, J., Wipf, D., and Philip, S. Y. From canonical correlation analysis to self-supervised graph neural networks. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Zhao, H., Yang, X., Wang, Z., Yang, E., and Deng, C. Graph debiased contrastive learning with joint representation clustering. In *Proc. IJCAI*, pp. 3434–3440, 2021.

Zheng, J., Wang, Y., Wang, G., Xia, J., Huang, Y., Zhao, G., Zhang, Y., and Li, S. Using context-to-vector with graph retrofitting to improve word embeddings. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8154–8163, Dublin, Ireland, May 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.acl-long.561.

Zhu, H., Sun, K., and Koniusz, P. Contrastive laplacian eigenmaps. *Advances in Neural Information Processing Systems*, 34, 2021a.

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., Liu, Q., and Wu, S. An empirical study of graph contrastive learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021b. URL https://openreview.net/forum?id=UuUbIYnHKO.

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

Zhu, Y., Xu, Y., Cui, H., Yang, C., Liu, Q., and Wu, S. Structure-Enhanced Heterogeneous Graph Contrastive Learning. In *Proceedings of the 2022 SIAM International Conference on Data Mining*, pp. 82–90. SIAM, 2022. doi: 10.1137/1.9781611977172.10. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611977172.10.

Appendix of ProGCL

A. Datasets

Table 7. Statistics of datasets used in experiments.

Dataset	Task	Nodes	Edges	Features	Classes
Amazon-Photo	Transductive	7,650	119,081	745	8
Amazon-Computers	Transductive	13,752	245,861	767	10
Coauthor-CS	Transductive	18,333	81,894	6,805	15
Wiki-CS	Transductive	11,701	216,123	300	10
Flickr	Inductive	89,250	899,756	500	7
Reddit	Inductive	231,443	11,606,919	602	41
Ogbn-arXiv	Inductive	169,343	1,166,243	128	40

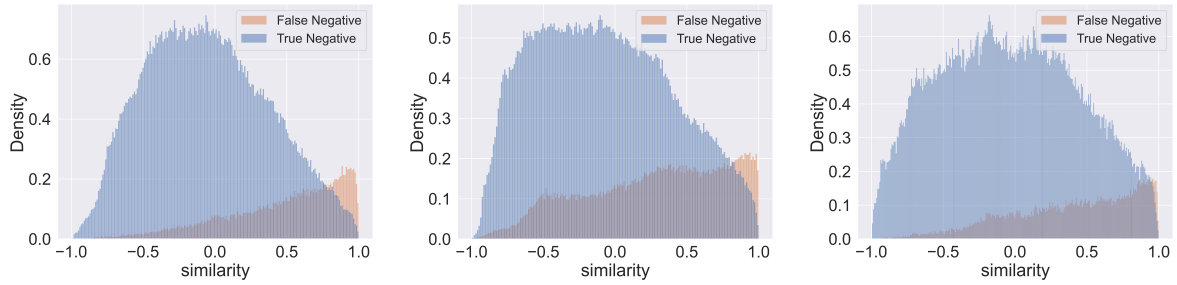
We introduce the datasets used in our experiments as follows:

- **WikiCS** (Mernyei & Cangea, 2020) is a reference network constructed based on Wikipedia. The nodes correspond to articles about computer science and edges are hyperlinks between the articles. Nodes are labeled with ten classes each representing a branch of the field. Node features are calculated as the average of pre-trained GloVe (Pennington et al., 2014) word embeddings of words in each article.
- **Amazon-Computers** and **Amazon-Photo** (Shchur et al., 2018) are two networks of co-purchase relationships constructed from Amazon, where nodes are goods and two goods are connected when they are frequently bought together. Each node has a sparse bag-of-words feature encoding product reviews and is labeled with its category.
- **Coauthor-CS** (Shchur et al., 2018) is an academic networks, which contain co-authorship graphs based on the Microsoft Academic Graph from the KDD Cup 2016 challenge. In the graph, nodes represent authors and edges indicate co-authorship relationships; that is, two nodes are connected if they have co-authored a paper. Each node has a sparse bag-of-words feature based on paper keywords of the author. The label of an author corresponds to their most active research field.
- **Flickr** originates from NUS-wide[1]. The SNAP website[2] collected Flickr data from four different sources including NUS-wide, and generated an un-directed graph. One node in the graph represents one image uploaded to Flickr. If two images share some common properties (e.g., same geographic location, same gallery, comments by the same user, etc.), there is an edge between the nodes of these two images. The node features are the 500-dimensional bag-of-word representation of the images provided by NUS-wide. For labels, we scan over the 81 tags of each image and manually merged them to 7 classes. Each image belongs to one of the 7 classes.
- **Reddit** (Hamilton et al., 2017) contains Reddit posts created in September 2014, where posts belong to different communities (subreddit). In the dataset, nodes correspond to posts, and edges connect posts if the same user has commented on both. Node features are constructed from post title, content, and comments, using off-the-shelf GloVe word embeddings, along with other metrics such as post score and the number of comments.
- **Ogbn-arXiv** is a large-scale graph from the Open Graph Benchmark (Hu et al., 2020). This is another citation network, where nodes represent CS papers on arXiv indexed by the Microsoft Academic Graph (Sinha et al., 2015). In our experiments, we symmetrize this graph and thus there is an edge between any pair of nodes if one paper has cited the

[1]https://lms.comp.nus.edu.sg/research/NUS-WIDE.htm
[2]https://snap.stanford.edu/data/web-flickr.html

other. Papers are classified into 40 classes based on arXiv subject area. The node features are computed as the average word-embedding of all words in the paper, where the embeddings are computed using a skip-gram model (Mikolov et al., 2013) over the entire corpus.

B. More Similarity Histograms of Negativess in SimCLR (CL) and GCA (GCL).



(a) Amazon-photo (Graph) (b) Amazon-computers (Graph) (c) WikiCS (Graph)

Figure 9. More similarity histograms of negative samples in and GCL.

As demonstrated in Figure 9, the negatives distribution over similarity varies significantly across GCL and CL. These phenomena provide explainations for the failure of existing techniques that emphasize hard negatives in GCL. More specifically, they regard the negatives that are most similar to anchor points as hard ones, which is feasible in CL. However, for graph-structured data (see Figure 9(a), Figure 9(b) and Figure 9(c)), many "hard" ones are false negatives indeed, which will undesirably push away the semantically similar samples.

C. More Experimental Details

C.1. Transductive learning

We adopt a two-layer GCN (Kipf & Welling, 2016a) as the encoder for transductive learning following previous works (Velickovic et al., 2019; Zhu et al., 2020). We can describe the architecture of the encoder as,

$$\begin{aligned} \mathrm{GC}_i(\boldsymbol{X}, \boldsymbol{A}) &= \sigma\left(\hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{X}\boldsymbol{W}_{\boldsymbol{i}}\right), \\ f(\boldsymbol{X}, \boldsymbol{A}) &= \mathrm{GC}_2\left(\mathrm{GC}_1(\boldsymbol{X}, \boldsymbol{A}), \boldsymbol{A}\right), \end{aligned} \tag{17}$$

where $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ is the adjacency matrix with self-loops and $\hat{\boldsymbol{D}} = \sum_i \hat{\boldsymbol{A}}_i$ is the degree matrix, $\sigma(\cdot)$ is a nonlinear activation function. $\boldsymbol{W_i}$ is the learnable weight matrix.

C.2. Inductive learning

Considering the large scale of some graph datasets, we adopt a three-layer GraphSAGE-GCN with residual connections (He et al., 2016) as the encoder following DGI (Velickovic et al., 2019) and GRACE (Zhu et al., 2020). The architecture of the encoder can be formulated as,

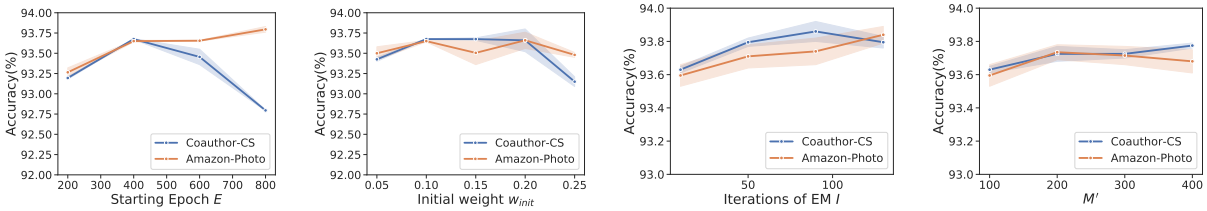
$$\begin{aligned} \widehat{\mathrm{MP}}_i(\boldsymbol{X}, \boldsymbol{A}) &= \sigma\left(\left[\hat{\boldsymbol{D}}^{-1}\hat{\boldsymbol{A}}\boldsymbol{X}; \boldsymbol{X}\right]\boldsymbol{W}_i\right), \\ f(\boldsymbol{X}, \boldsymbol{A}) &= \widehat{\mathrm{MP}}_3\left(\widehat{\mathrm{MP}}_2\left(\widehat{\mathrm{MP}}_1(\boldsymbol{X}, \boldsymbol{A}), \boldsymbol{A}\right), \boldsymbol{A}\right). \end{aligned} \tag{18}$$

D. Hyper-parameters Analysis

Following GRACE (Zhu et al., 2020) and GCA (Zhu et al., 2021c), we initialize the model parameters with Glorot initialization (Glorot & Bengio, 2010) and train the model using Adam SGD optimizer for all datasets. The ℓ_2 weight decay factor is seted as 10^{-5} and the dropout rate is seted to zero. The parameters which control the sampling process of two views are the

Table 8. Hyperparameters specifications. "Starting epoch" E refers to the epoch for fitting BMM; Initial weight w_{init} is the initial weight for false negative component of mixture distribution; "Iterations" is the iterations I of EM algorithm.

Dataset	τ	learning rate	Training epochs	Hidden dimension	Activation function	Starting epoch	Initial weight	Iterations
Amazon-Photo	0.3	0.01	2500	128	RRelu	400	0.15	10
Amazon-Computers	0.2	0.01	2000	128	RRelu	400	0.05	10
Coauthor-CS	0.2	0.0001	1000	256	RRelu	400	0.05	10
Wiki-CS	0.4	0.01	4000	256	PRelu	50	0.05	10
Reddit	0.4	0.0001	80	512	ELU	40	0.01	10
Filckr	0.4	0.0001	80	512	ELU	20	0.15	10
Ogbn-arXiv	0.4	0.0001	100	512	ELU	20	0.02	10



(a) E (b) w_{init} (c) I (d) M'

Figure 10. Accuracy when varying E, w_{init}, I and $M'(M = NM')$ for BMM.

same as GRACE and GCA. Other hyper-parameters of ProGCL can be seen in Table 8. For transductive task, the two hyper-parameters were chosen in a grid $E \in \{50, 100, 200, 300, 400, 600, 800\}$ and $w_{init} \in \{0.01, 0.05, 0.10, 0.15, 0.20, 0.25\}$. We further study the influence of E, w_{init}, I and M in Figure 10. Firstly, the accuracy varies significantly across various starting epoch E in Amazon-Photo while varies slightly in Coauthor-CS, which illustrates the importance of tuning E varies across datasets. However, generally speaking, ProGCL's performance does not see sharp drop when varying E. This validates that BMM is flexible enough to fit various distributions of different epochs. As shown in figure 10(b), the initial weight w_{init} does not influence much, which illustrates that BMM can split the two component well to an ideal ratio even if the initial weight is far from intuitive one (the reciprocal of classes number). As shown in figure 10(c), we can observe minor improvement when we iterate EM algorithm more times. However, this will introduce more computational overhead and thus we set $I = 10$ in all experiments for convenience. As shown in Figure 9(c), sampling more similarities for fitting the BMM can bring minor improvements, however, it will introduce much more computational overhead. In our experiments, we only sample $M' = 100$ samples for each anchor point. Thus, the number of total selected samples $M = NM'$.

E. Pseudo Codes of Inductive Learning.

Different from transductive learning, it is not feasible to compute all the pairwise similarities for large-scale graph datasets. Thus, we extend ProGCL to inductive setting. The algorithm of ProGCL for inductive learning can be seen as follows. The equations mentioned in the algorithm can be seen in the main text.

Algorithm 2 ProGCL-weight & ProGCL-mix (Inductive)

Input: $\mathcal{T}, \mathcal{G}, f, g, N$, normalized cosine similarity s, epoch for fitting BMM E, $mode$ ('weight' or 'mix'), empty list $\mathcal{P}$ for storing the estimated probabilities, Batchsize B.
for $epoch = 0, 1, 2, ...$ **do**
  $k = 0$;
  for each mini-batch **do**
    $\mathcal{G}_s\,(\mathcal{V}_s, \mathcal{E}_s) \leftarrow$ Sampled sub-graph of $\mathcal{G}$ with sampling rules of GraphSAGE;
    Draw two augmentation functions $t \sim \mathcal{T}$, $t' \sim \mathcal{T}$;
    $\widetilde{\mathcal{G}}_s^{(1)} = t(\mathcal{G}_s)$, $\widetilde{\mathcal{G}}_s^{(2)} = t'(\mathcal{G}_s)$;
    $\mathcal{U}_s = f(\widetilde{\mathcal{G}}_s^{(1)})$, $\mathcal{V}_s = f(\widetilde{\mathcal{G}}_s^{(2)})$;
    for all $\boldsymbol{u}_i \in \mathcal{U}_s$ and $\boldsymbol{v}_i \in \mathcal{V}_s$ **do**
      $s_{ij} = s(g(\boldsymbol{u}_i), g(\boldsymbol{v}_i))$;
      if $epoch = E$ **then**
        Compute $\mathcal{M}_{i,j} = p\left(c_t \mid s_{ij}\right)$ with Eq. (4) to Eq. (9);
        $\mathcal{P}$.append($\mathcal{M}$).
      end if
      if $epoch \geq E$ **then**
        if $mode$ = 'weight' **then**
          Compute $\mathcal{J}_w$ with Eq. (10) to Eq. (12); ($\mathcal{P}$[k] as the estimated probabilities for k-th minibatch)
          Update the parameters of f, g with $\mathcal{J}_w$.
        end if
        if $mode$ = 'mix' **then**
          Compute $\mathcal{J}_m$ with Eq. (13) to Eq. (16); ($\mathcal{P}$[k] as the estimated probabilities for k-th minibatch)
          Update the parameters of f, g with $\mathcal{J}_m$.
        end if
      else if $epoch < E$ **then**
        Compute $\mathcal{J}$ with Eq. (1) to Eq. (2);
        Update the parameters of f, g with $\mathcal{J}$.
      end if
    end for
    $k = k + 1$.
  end for
end for
Output: f, g.