DropMix: Better Graph Contrastive Learning with Harder Negative Samples

Yueqi Ma

Minjie Chen

Xiang Li*

School of Data Science and EngineeringSchool of Data Science and EngineeringSchool of Data Science and EngineeringEast China Normal UniversityEast China Normal UniversityEast China Normal UniversityShanghai, ChinaShanghai, ChinaShanghai, China51215903110@stu.ecnu.edu.cnminjiechen@stu.ecnu.edu.cnxiangli@dase.ecnu.edu.cn

Abstract—While generating better negative samples for contrastive learning has been widely studied in the areas of CV and NLP, very few work has focused on graph-structured data. Recently, Mixup has been introduced to synthesize hard negative samples in graph contrastive learning (GCL). However, due to the unsupervised learning nature of GCL, without the help of soft labels, directly mixing representations of samples could inadvertently lead to the information loss of the original hard negative and further adversely affect the quality of the newly generated harder negative. To address the problem, in this paper, we propose a novel method DropMix to synthesize harder negative samples, which consists of two main steps. Specifically, we first select some hard negative samples by measuring their hardness from both local and global views in the graph simultaneously. After that, we mix hard negatives only on partial representation dimensions to generate harder ones and decrease the information loss caused by Mixup. We conduct extensive experiments to verify the effectiveness of DropMix on six benchmark datasets. Our results show that our method can lead to better GCL performance. Our data and codes are publicly available at https://github.com/Mayueq/DropMix-Code.

Index Terms—Graph neural network, Contrastive learning, Hard sample mining

I. INTRODUCTION

With the explosion of data volume in the real world, labeled data is generally hard to derive, while more available data is unlabeled. To leverage massive unlabeled data, self-supervised learning (SSL) [1], [2] has recently received increasing attention. Contrastive learning (CL) [3], a mainstream type of SSL, has been widely used in the fields of CV [4], [5] and NLP [6], [7]. The major goal of CL is to maximize the similarity between positive pairs while minimizing that between negative ones. CL has also been extended to graphs and integrated with graph neural networks (GNNs) to learn node/graph representations, whose superior performance has been verified [8].

Some previous studies [9], [10] have shown that the key to affecting the performance of contrastive learning is the selection of positive and negative samples. Generally, positive samples are constructed by data augmentation methods [11], [12], while negative samples are randomly selected. Recently, it has been pointed out in [13] that harder negative samples are more helpful for learning. Therefore, there has been a growing



Fig. 1. Three different Mix methods to synthesize harder negative samples. Each square represents one dimension.

interest in selecting harder negative samples for contrastive learning [14], [15]. Further, negative samples can also be constructed by artificial synthesis. For example, Mixup [16] and CutMix [17] are two data augmentation methods that have been applied in the area of CV, which mix different images and their labels to synthesize new samples with soft labels. The difference is that Mixup performs interpolation between two images, while CutMix cuts a patch from an image and pastes a new one from another image. However, very few of existing studies focuses on graph-structured data. In addition, these methods heavily depend on the information of soft labels and cannot be directly applied in the unsupervised learning settings.

To address these problems, a recently proposed method ProGCL [18] first selects hard negatives and then uses Mixup [16] to generate harder ones in graph contrastive learning (GCL), which is essentially unsupervised learning. Despite the success, it has two major problems. On the one hand, it measures the hardness of negative samples based on only their local information in the graph, while the global information contained in the graph can further provide a supplementary view for hard negative selection. On the other hand, it performs Mixup on all the dimensions of hard negative representations. However, since the selected negative is already a hard one, this could inadvertently lead to the information loss of the original sample and adversely affect the quality of the newly synthesized negative.

^{*} Corresponding author.

In this paper, to tackle these issues, we propose a novel Mixup-based method DropMix for synthesizing harder negative samples in GCL. Our goal is to increase the hardness of negative samples without the supervision of soft labels in the unsupervised learning setting, while decreasing the information loss of the original hard negative samples caused by Mixup. Specifically, we first measure the hardness of negative samples by computing their similarities with positive samples from both local and global views of information. For the local view, we consider the neighborhood of a node. For the global view, we employ the diffusion matrix [19] of the graph. Then we rank the similarities and select hard negatives. After that, inspired by CutMix, we mix these selected hard negatives only in partial dimensions to construct new harder negatives. while keeping the remaining part unchanged. This explains the origination of Drop in DropMix. In this way, we can alleviate the information loss of the original hard negatives, even without the help of soft labels. Figure 1 illustrates the difference between Mixup, CutMix, and our proposed method DropMix. Finally, our main contributions in this paper are summarized as follows:

- We present a novel method DropMix for synthesizing harder negative samples in GCL under the unsupervised learning setting.
- We unify the local and global views of information in the graph to measure the hardness of negative samples, which can be used to effectively filter easy negatives and false negatives.
- We propose to perform Mixup only on a proportion of dimensionalities of the hard negative samples, to decrease the information loss caused by Mixup.
- We conduct extensive experiments to show the superiority of our proposed method in generating harder negative samples in GCL.

II. RELATED WORK

A. Graph Contrastive Learning

In recent years, inspired by contrastive learning in CV and NLP, more and more studies have begun to apply contrastive learning to graphs and have obtained inspiring results as well. As an unsupervised learning method, GCL has achieved comparable or even better results than many supervised learning methods, without relying on the involvement of a large number of labels, which greatly improves the utilization efficiency of data in the real world. For example, a representative model DGI [20] first proposes to maximize the mutual information between node-level and graph-level representations to learn node representations. Based on DGI, MVGRL [21] proposes to use node diffusion method to learn node representations by maximizing mutual information between node-level and graph-level representations of the original and diffusion graph. Further, GMI [22] develops an unsupervised learning model trained by maximizing the correlation between input graphs and high-level hidden representations. GraphCL [12] explores the impact of different data augmentation methods on different

domain datasets. BGRL [23] learns a node representation by encoding two augmented versions of a graph using two distinct graph encoders. Besides, MERIT [24] learns node representations by enhancing Siamese self-distillation with multi-scale contrastive learning. In addition, Grace [25] and GCA [26] are jointly consider corruption at both topology and node attribute levels to provide diverse contexts for nodes in different views.

B. Negative Sampling

Since the goal of contrastive learning is to pull the positive samples close to the anchor and push away the negative samples, how to construct positive and negative samples is particularly important. Some existing works use data augmentation methods [11], [12] to generate better and more representative positive samples, while the studies on generating negative samples are insufficient in comparison. Most recent studies on negative sampling focus on how to select better negative samples from negative sample sets. For example, HCL [14] first uses sampling distribution to generate negative samples and then presents a sampling strategy on sampling in the absence of real dissimilar information. What's more, KGPolicy [15] considers both the similarities between the negative and the anchor (the positive samples). After that, the two similarities are combined as the measure for selecting hard negative samples. For graph-structured data, some works such as [27] propose to cluster nodes in the training process, and generate pseudo-labels for nodes based on the clustering results to decide which negative samples to select. Further, CuCo [28] introduces Curriculum Learning and designs a scoring function for the negative samples, ranking them from easy to difficult and learning them sequentially.

C. Mixup

Mixup [16] is an effective method for image data augmentation. It interpolates between two images in proportion to generate a new sample and also proportionally mixes their labels to get the corresponding soft label. Some recent studies have applied Mixup to graph data augmentation. For example, a recent work [29] applies Mixup to both node classification and graph classification tasks with GNNs. GraphMix [30] proposes a data augmentation method for training fully connected networks jointly with GNNs through parameter sharing and interpolation-based regularization. Additionally, Mixup has also been used for the synthesis of hard negative samples recently. MoCHi [31] first proposes to introduce Mixup into the construction of negative samples by mixing hard negative samples or positive samples with hard negative samples to synthesize harder negative samples artificially. And on graphs, ProGCL [18] and M-Mix [32] similarly uses Mixup for the synthesis of hard negative samples.

D. CutMix

CutMix [17] is another data augmentation method that combines both Dropout and Mixup. It first cuts a patch from an image and then pastes a new one from another



Fig. 2. (left) First step, we measure the hardness of negative samples in the original graph with local information and the diffusion graph with global information. The top and bottom graphs represent node v_1 , v_2 and a subgraph structure around them respectively. (right) Second step, we construct new harder negative samples by mixing two hard negative samples in some dimensions. Specifically, Mix and \odot denote eq. (7) and mask respectively, which represent we mix h_1 and h_2 to generate green dimensions by mixing blue and yellow while retaining some yellow dimensions.

image to obtain new ones. After that, it mixes the ground truth labels proportionally according to the area size of the patch. In this way, CutMix can alleviate the problem that the samples generated by Mixup tend to be unnatural. However, this method is mainly developed for images but not for graphstructured data.

Despite the success, through experiments, we observe that the performances of hard negatives generated by Mixup and CutMix are not as good as we expected in the unsupervised learning settings. Different from them, DropMix can generate better and harder negative samples without labels.

III. METHOD

A. Problem Definition

Let $G = (\mathcal{V}, \mathcal{E})$ denote an undirected graph, where $\mathcal{V} = \{v_1, v_2...v_N\}$ is a set of N nodes and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ denotes the adjacency relationships between nodes in \mathcal{V} . Additionally, node feature $\mathcal{X} \in \mathbb{R}^{N \times d}$ is a d-dimensional matrix, and each node v_i is associated with a feature vector x_i . The graph Gcan be described as an adjacency matrix $A \in \mathbb{R}^{N \times N}$ according to \mathcal{E} , so that we can compute the symmetrically normalized adjacency matrix \hat{A} by $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, where \tilde{A} is the adjacency matrix with added self-loops, and \tilde{D} is the diagonal degree matrix of \tilde{A} .

B. Graph Encoder

In this section, we use Graph Neutral Networks (GNNs) as our encoder to learn the nodes' representations through the message-passing scheme. For any node v_i in the *l*-th layer, we can obtain its embedding by aggregating the information in the (l-1)-th layer from itself and all its neighbors:

$$h_u^{(l)} = \operatorname{AGGREGATE}\left(h_u^{(l-1)} : u \in \mathcal{N}(i)\right), \quad (1)$$

$$h_i^{(l)} = \text{COMBINE}\left(h_i^{(l-1)}, h_u^{(l)}\right), \tag{2}$$

where $h_i^{(l)}$ is the representation vector of node v_i in the l-th layer with $h_i^{(0)} = x_i$. Further, AGGREGATE(·) and COMBINE(·) are aggregation function and combination function of the GNN layer, respectively, and $\mathcal{N}(i)$ denotes the neighbor set of node v_i .

C. Graph Contrastive Learning

To achieve the goal of graph contrastive learning, i.e., minimizing the distance between the anchor and the positive samples, while maximizing that between the anchor and the negative samples, we adopt the InfoNCE loss function [33]. Specifically, given a node v_i and its positive sample v_j which can be obtained by data augmentation, we can derive their embeddings h_i and h_j by GNN encoders. And then we take all the other nodes in the graph as negative samples. Formally, the training objective for the positive pair (h_i, h_j) is defined as:

$$L_{i,j} = -\log \frac{\exp(\sin(h_i, h_j/\tau))}{\sum_{k=1}^{N} \exp(\sin(h_i, h_k/\tau))},$$
(3)

where τ denotes the temperature parameter and sim(·) represents the similarity function between h_i and h_j .

D. Hard Negative Selection

In this section, we select hard negative samples from the negative sample set, which consists of nodes other than the positive. And details are given in the left part of figure 2. First, we consider that it may not enough to measure the degree of hardness of negative samples only from the local view of the original graph. Therefore, inspired by graph diffusion [19], we transform the adjacency matrix into a diffusion matrix, which can represent the global connections between two nodes. After that, we measure the similarities between negative samples and positive samples from both the local and global views. Finally, we combine the two similarities to derive the measurement for the degree of hardness w.r.t. a given negative sample.

Similar as in [19], we define generalized graph diffusion that is computed by using fast approximation and sparsification methods in eq. (4). Note that $T \in \mathbb{R}^{N \times N}$ is the generalized transition matrix and θ_k is the weighting coefficients, requiring $\sum_{k=0}^{\infty} \theta_k = 1, \ \theta_k \in [0, 1]$ and $\lambda_i \in [0, 1]$, where λ_i are eigenvalues of T.

$$S = \sum_{k=0}^{\infty} \theta_k T^k.$$
(4)

In our work, we use Personalized PageRank (PPR) [34], which is one of the instantiations of generalized graph diffusion. Specifically, we set $T = \hat{A}$ and $\theta_k = \alpha (1 - \alpha)^k$, where α denotes teleport probability in a random walk. The closedform solution to PPR diffusion is formulated as:

$$S = \alpha (I_n - (1 - \alpha)\hat{A})^{-1}.$$
 (5)

In this way, we have constructed a new matrix S that reflects the global relations between nodes in the graph. Then we can measure the hardness of negative samples by computing their similarities to the positive samples in the views of both the adjacency matrix and S. For simplicity, we choose cosine similarity as the measure function. Specifically, given a node v and its positive sample v_p , for any negative sample v_n , we use Φ_l and Φ_g to denote the hardness between v_p and v_n in the local and global views, respectively:

$$\Phi_{l} = \frac{h_{p}^{T}h_{n}}{\|h_{p}\|\cdot\|h_{n}\|}, \Phi_{g} = \frac{h_{p}^{T}h_{n}}{\left\|\tilde{h}_{p}\right\|\cdot\left\|\tilde{h}_{n}\right\|},$$
(6)

where h_p, h_n, h_p, h_n are the embeddings derived by the GNN encoder in section III-B. Here, h_p, h_n denote the embeddings of positive and negative samples in the original graph, while \tilde{h}_p, \tilde{h}_n represent that in the diffusion graph. After that, we combine the two measures to get the final degree of hardness $\Phi = \Phi_l + \Phi_g$ and then select hard negatives. Note that the small hardness value generally corresponds to easy negative samples that are useless, while the large one indicates false negative samples that may hurt the model performance. Therefore, we rank negative samples based on Φ , remove samples at the two ends, and select the remaining samples as hard negatives. Specifically, we set two hyper-parameters α and β to control the lower and upper limit of the hardness of hard negatives respectively.

TABLE I STATISTICS OF DATASETS

Datasets	Nodes	Edges	Attributes	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
Wiki-CS	11,701	216,123	300	10
Amazon-Photo	7,650	119,081	745	8
Coauthor-CS	18,333	81,894	6,805	15

E. Hard Negative Mixing

After selecting hard negative samples from the negative sample set, we next mix the selected hard negatives to generate harder ones. The right part of figure 2 shows a toy example on DropMix. Let h_1, h_2 denote the representations of two hard negative samples, respectively. We first define the mixing operation for h_1 and h_2 as:

$$h_{mix} = \lambda h_1 + (1 - \lambda)h_2, \tag{7}$$

where $\lambda \in [0, 1]$ is the mixing coefficient which decides the mixing ratio. Then, we mask a proportion of dimensions with a binary mask vector M indicating which dimensions should be dropped out. In our work, we specify the proportion of one-valued entries in M as a hyper-parameter to control the number of dimensions selected properly. Finally, we can obtain a new negative sample by combining eq. (7) and the mask operation:

$$h_{new} = M \odot h_1 + (1 - M) \odot h_{mix}.$$
(8)

Based on eq. (8), we can generate new harder negative samples to help training with a novel mix method.

IV. DISCUSSION

In this section, we summarize the relationships between DropMix and two other data augmentation methods for hard negative generation: Mixup and CutMix. First, as shown in previous works [16], [17] both Mixup and CutMix perform well in the supervised learning settings, which require samples with labels. However, in the unsupervised learning settings, their performances could be degraded due to the lack of soft labels. Further, Mixup performs interpolation over all the embedding dimensions of samples and could inadvertently lead to the information loss of the original hard negative sample. For CutMix, although it can reduce the information loss by keeping some parts in the original negative unchanged, it discards other parts of the sample and loses the corresponding information completely. Different from Mixup and CutMix, DropMix mixes the representations of samples only in partial dimensions and keeps others unchanged, which decreases the information loss in the original negative sample. This further weakens the need on soft labels and extends its applicability in the unsupervised learning settings. Technically, DropMix subsumes both Mixup and CutMix.

 TABLE II

 The classification accuracy (%) over the methods on 6 datasets. "OOM" denotes out of memory on a 32GB GPU. The highest performances are highlighted in bold.

Methods	Cora	Citeseer	Pubmed	Wiki-CS	Amazon-Photo	Coauthor-CS
GCN	81.80 ± 0.50	70.80 ± 0.50	79.30 ± 0.70	77.19 ± 0.12	92.42 ± 0.22	93.03 ± 0.31
GAT	83.00 ± 0.70	72.50 ± 0.70	79.00 ± 0.30	77.65 ± 0.11	92.56 ± 0.35	92.31 ± 0.24
GAE	71.55 ± 0.33	65.87 ± 0.42	72.15 ± 0.50	70.15 ± 0.01	91.62 ± 0.13	90.01 ± 0.17
VGAE	73.27 ± 0.47	66.92 ± 0.50	74.13 ± 0.62	75.35 ± 0.14	92.20 ± 0.11	92.11 ± 0.09
DGI	83.80 ± 0.50	72.00 ± 0.60	77.90 ± 0.30	75.35 ± 0.14	91.61 ± 0.22	92.15 ± 0.63
GMI	83.00 ± 0.30	72.40 ± 0.10	79.90 ± 0.20	74.85 ± 0.08	90.68 ± 0.17	OOM
MVGRL	86.80 ± 0.50	73.30 ± 0.50	80.10 ± 0.70	77.43 ± 0.17	92.08 ± 0.01	92.18 ± 0.05
BGRL	84.68 ± 0.23	73.88 ± 0.15	80.73 ± 0.17	78.41 ± 0.09	92.95 ± 0.07	92.72 ± 0.03
MERIT	83.10 ± 0.60	74.00 ± 0.70	80.10 ± 0.40	78.35 ± 0.05	92.53 ± 0.15	92.51 ± 0.14
ProGCL	83.50 ± 0.22	74.19 ± 0.13	80.77 ± 0.15	78.45 ± 0.04	93.64 ± 0.13	93.67 ± 0.12
MVGRL+Mixup	86.92 ± 0.44	74.08 ± 0.28	80.62 ± 0.52	78.23 ± 0.46	93.55 ± 0.63	94.66 ± 0.21
MVGRL+CutMix	86.87 ± 0.66	74.24 ± 0.52	80.68 ± 0.40	78.42 ± 0.64	93.47 ± 0.39	95.21 ± 0.50
MVGRL+DropMix	87.17 ± 0.31	74.74 ± 0.23	81.29 ± 0.26	78.82 ± 0.16	94.46 ± 0.33	96.66 ± 0.52

V. EXPERIMENTS

In this section, we evaluate DropMix's effectiveness. We compare DropMix with 10 other methods by their accuracies on node classification tasks. We also analyze the performance of each component of DropMix by ablation study and hyperparameter sensitivity analysis. Further, we verify that DropMix can be used in different GCL models and it is very suitable for graph-structured data.

A. Datasets

We use six datasets which are widely used for node classification tasks to verify the performance of DropMix including Cora, Citeseer, Pubmed, Amazon-Photo, Wiki-CS and Coauthor-CS. The first three are citation networks, where each node represents a document and each edge is a citation link. Amazon Photo is a segment of the Amazon co-purchase graph, where nodes represent goods and edges indicate that two goods are frequently bought together. Wiki-CS is a Wikipedia-based dataset, which consists of nodes corresponding to Computer Science articles, with edges based on hyperlinks and 10 classes representing different branches of the field. Coauthor CS is a co-authorship graph based on the Microsoft Academic Graph from the KDD Cup 2016 challenge. Here, nodes are authors, which 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. The specific statistics are summarized in Table I.

B. Experimental Setting

We implement DropMix using PyTorch 1.10.0. We adopt the task of node classification to evaluate the performance of node representation learning. Further, We adopt MVGRL [21] as the base model and use test accuracy as the evaluation indicator to demonstrate the effectiveness of DropMix. We share the GNN encoder in the two views of local and global. For DropMix

and all its variants, we set the learning rate to 0.001 on Cora, Citeseer and Pubmed, 0.0007 on the other, and the penalty weight on the l_2 -norm regularizer to 0.002 on Citeseer. We use early stopping with the patience of 40 on Amazon-photo, Coauthor-CS and Wiki, and 20 on the other datasets, i.e., we stop training if the validation accuracy does not decrease for 40 or 20 consecutive epochs. We fine-tune the model hyperparameters by grid search. Specially, we set γ as the rate of representation dimensions $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$, and the mixup rate λ {0.1, 0.2, 0.3, 0.4}. What's more, to the lower limit proportion of the hardness of hard negatives, we fine-tune it from 5% to 50% with a 5% increment each time. And to the upper limit proportion, we fine-tune it from $\{80\%, 85\%,$ 90%, 95% }. For a fair comparison, we run all the experiments on a server with 32G memory and a single NVIDIA 2080Ti GPU. Additionally, for all the baseline methods, we use the original results released by their papers, and for some methods whose other experimental results on some datasets can not be found, we use the records which are reported in [18] and [24]. However, for the latest work ProGCL, we use the default parameters reported in the original paper and the original codes to run the experiment on Cora, Citeseer and Pubmed. For each method, we run experiments 10 times and report the average results.

C. Comparison with the State-of-the-Arts

To demonstrate the effectiveness of DropMix, we first evaluate it with 10 state-of-the-art methods which can be grouped into two categories on six datasets. Specifically, we choose multiple baselines which are unsupervised learning methods including Graph AutoEncoders(GAE, VGAE) [35], DGI [20], GMI [22], MVGRL [21], BGRL [23], MERIT [24] and ProGCL [18] which is the most advanced method introducing Mixup into the synthesis of negative samples as introduced in the related work. What's more, we also compare



Fig. 3. The results of different mix methods on three datasets.

TABLE III The results of different methods based GCA , the best performances are highlighted in bold.

Datasets	Cora	Citeseer	Amazon-Photo
GCA	80.90 ± 0.41	72.14 ± 0.06	92.55 ± 0.03
ProGCL	83.50 ± 0.22	74.19 ± 0.13	93.64 ± 0.13
GCA+DropMix	83.82 ± 0.25	74.95 ± 0.20	94.13 ± 0.11

DropMix with supervised learning methods including GCN [8], and Graph Attention Network (GAT). GCN is a model that extends the convolution operation on graphs. And Graph Attention Network further integrates the attention mechanism in the convolutional layer. In addition, we use only Mixup or CutMix, respectively, in the same unsupervised learning setting to generate new negative samples as the methods we compared.

Table II summarizes the performance results and the best results are in bold. We can observe that the test accuracy of DropMix outperforms all the other methods on six datasets. This indicates that we can improve the performance of a GCL model significantly only by supplying high-quality negative samples. Compared with ProGCL, which also uses Mixup to construct new negative samples, DropMix selects better hard negative samples by considering local information and global information simultaneously. Furthermore, it can be seen from the table that, although Mixup and CutMix can both bring some degree of improvement to the base model, the performance of DropMix is better. This is because it only mixes parts of dimensions to generate new negative samples so that less original information of the hard negative samples can be lost.

D. Results on different GCL Models

We evaluate the performance of DropMix not only on the method of MVGRL, but on GCA [26] to get a fairer comparison with ProGCL which is based on GCA. Results are shown in Table III. The experimental results of GCA are taken from [18] and [36]. It can be seen from the table that

TABLE IV The ablation study of multi-view measure on three datasets , the best performances are highlighted in bold.

Datasets	Wiki-CS	Amazon-Photo	Coauthor-CS
DropMix-ol	78.53 ± 0.36	94.19 ± 0.55	95.62 ± 0.28
DropMix-og	78.71 ± 0.43	94.08 ± 0.30	96.32 ± 0.39
DropMix	78.82 ± 0.16	94.46 ± 0.33	96.66 ± 0.52

the performance is better than the base model GCA by adding DropMix on it. Besides this can demonstrate that DropMix can be used in different GCL methods to improve their performances by reducing information loss when constructing new hard negatives.

E. Ablation study

In this section, we conduct an ablation study on DropMix to further understand the characteristics of its main components. We consider different variants of the two steps to study the effect of multi-views measure and partial dimensions mixing respectively.

1) Effect of multi-views measure: In our method, how to select hard negatives is important. We consider that measuring the hardness of negative samples from both the local view and global view may lead to better performance. Therefore, we do the same operation as DropMix for three datasets, but only use local or global information respectively to measure. The results are shown in Table IV, where "DropMix-ol" means DropMix with only local view and "DropMix-og" means DropMix with only global view. We can observe that DropMix achieves better performance than both DropMix-ol and DropMix-og, and DropMix-og beats DropMix-ol in most cases. This shows that compared with only measuring the hardness by local information, global information may show better performance, because of the rich information it incorporates from the global view. However, it's obvious that the best way to measure is considering the two views at the same time.

2) Effect of partial dimensions mixing: We replace our method with the ordinary Mixup and CutMix on node rep-

TABLE V The results of different methods based MVGRL , and the highest performances are marked in bold.

Datasets	Citeseer	Amazon-Photo	
MVGRL	73.30 ± 0.50	92.08 ± 0.01	
Input Feature + Mixup Input Feature + CutMix Input Feature + DropMix	$\begin{array}{c} 73.60 \pm 0.78 \\ 73.89 \pm 0.66 \\ 73.82 \pm 0.42 \end{array}$	$\begin{array}{c} 92.82 \pm 0.45 \\ 92.56 \pm 0.30 \\ 92.76 \pm 0.50 \end{array}$	
Node Representation + Mixup Node Representation + CutMix Node Representation + DropMix	$\begin{array}{l} 74.08 \pm 0.19 \\ 74.24 \pm 0.37 \\ \textbf{74.74 \pm 0.23} \end{array}$	$\begin{array}{l} 93.55 \pm 0.41 \\ 93.47 \pm 0.42 \\ \textbf{94.46} \pm \textbf{0.33} \end{array}$	



Fig. 4. Parameter analysis of the hardness of hard negative samples on Wiki-CS.

resentations to study the impact of partial dimensions mixing. Mixup denotes we mix all the dimensions at the same time, and CutMix denotes we change some dimensions of a negative sample by the corresponding dimensions of another one. We evaluate the performance of the three methods and report the performance in Figure 3. We can observe that Mixup and CutMix can bring some degree of improvement. Nevertheless, DropMix clearly outperforms them additionally on three datasets. This indicates that our method which mixes node representations on partial dimensions achieves success in GCL, and further shows the importance of reducing hard information loss while generating new hard negatives.

F. Effect of DropMix on Input Feature and Node Representation

Both Mixup and CutMix are data augmentation methods that are usually used in the input layer. To analyze the performance of DropMix on graph-structured data to synthesize negative samples, we show the results of DropMix and others implemented in the input layer and node representation generated by GNN, respectively. The results are shown in Table V. It can be seen that the performance of mixing on node representations is better than on the input feature, this may benefit from the graph structure information used in GNN. Further, we observe that in the input layer, the performance



Fig. 5. Parameter analysis of mix on Amazon-photo.

gap of DropMix against Mixup and CutMix is marginal, and even worse because of the sparse characteristic of graph data. However, when we use our method on node representations, DropMix presents its superiority on graph-structured data.

G. Hyper-parameters Sensitivity Analysis

We end this section with a sensitivity analysis of the hyperparameters. In particular, we study four key hyper-parameters in two steps: The lower and upper hardness limit α and β of hard negatives we select in the first step, the rate γ of dimensions to mix, and the rate λ of mixup. In our experiments, we vary one parameter each time with others fixed.

1) Number of hard negative samples: In section V-E, we have studied the effect of multi-views measurement. However, after measuring the hardness of negative samples, how to choose the most proper hardness range is also important. We use hyper-parameters α and β to control the lower and upper limit of the range, respectively. For example, α =35% and β =95% denote that we choose hard negative samples whose hardness values are ranked within 35%-95% among all the negative samples. And then, we use the most proper α and β in both DropMix and DropMix-ol to achieve the best performance respectively, where "DropMix-ol" means only using local information. Figure 4 shows the results on Wiki-CS. We can observe that too high or too low hardness both hurt the performance so it's essential to discard easy negatives that are useless and the hardest ones which may be false negatives. Additionally, it can be seen that through measuring from multiviews, we can not only select more precise hard negatives but get better results through fewer negatives.

2) Hyper-parameters of mix: We show the results of the two hyper-parameters on Amazon-Photo in Figure 5, where the scores on the horizontal axis denote the number of dimensions we randomly selected as a proportion of all dimensions. From this figure, we can see that both hyper-parameters can impact the performance of the method, where the rate of dimensions to mixup is more important. It's obvious that if γ is more than 0.5, the performance drops rapidly. Thus, it can prove that

it's necessary to decrease the information loss of the original negatives. However, a too small γ may lead to the mix being ineffective. This further shows that Dropmix which mixes on partial dimensions is an effective method for generating harder negative samples.

VI. CONCLUSION

In this paper, we are devoted to learning better representations for GCL. We propose a novel method to synthesize harder negative samples. Specifically, we present a two-step method to implement. We first select some hard negative samples by measuring their hardness from both local and global views in the graph simultaneously. Second, we mix hard negative samples only on partial dimensions to increase the hardness of negative samples and decrease the information loss caused by Mixup. Finally, experimental results demonstrate that our method performs favorably on six datasets.

ACKNOWLEDGMENT

This work is supported by Shanghai Pujiang Talent Program No. 21PJ1402900, Shanghai Science and Technology Committee General Program No. 22ZR1419900 and National Natural Science Foundation of China No. 62202172.

REFERENCES

- J. Neville and D. Jensen, "Iterative classification in relational data," in Proc. AAAI-2000 workshop on learning statistical models from relational data, 2000, pp. 13–20.
- [2] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the* 20th International conference on Machine learning (ICML-03), 2003, pp. 912–919.
- [3] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," Advances in neural information processing systems, vol. 32, 2019.
- [4] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.
- [5] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.
- [6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.
- [7] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.
- [8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
- [9] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," arXiv preprint arXiv:2003.04297, 2020.
- [10] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, "What makes for good views for contrastive learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 6827–6839, 2020.
- [11] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11015– 11023.
- [12] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5812–5823, 2020.
- [13] T. T. Cai, J. Frankle, D. J. Schwab, and A. S. Morcos, "Are all negatives created equal in contrastive instance discrimination?" *arXiv preprint* arXiv:2010.06682, 2020.

- [14] J. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka, "Contrastive learning with hard negative samples," *arXiv preprint arXiv:2010.04592*, 2020.
- [15] X. Wang, Y. Xu, X. He, Y. Cao, M. Wang, and T.-S. Chua, "Reinforced negative sampling over knowledge graph for recommendation," in *Proceedings of the web conference 2020*, 2020, pp. 99–109.
- [16] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization (2017)," arXiv preprint arXiv:1710.09412, 2017.
- [17] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF international conference on computer* vision, 2019, pp. 6023–6032.
- [18] J. Xia, L. Wu, G. Wang, J. Chen, and S. Z. Li, "Progcl: Rethinking hard negative mining in graph contrastive learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 24332–24346.
- [19] J. Klicpera, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," arXiv preprint arXiv:1911.05485, 2019.
- [20] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax." *ICLR (Poster)*, vol. 2, no. 3, p. 4, 2019.
- [21] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4116–4126.
- [22] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, "Graph representation learning via graphical mutual information maximization," in *Proceedings of The Web Conference 2020*, 2020, pp. 259– 270.
- [23] S. Thakoor, C. Tallec, M. G. Azar, R. Munos, P. Veličković, and M. Valko, "Bootstrapped representation learning on graphs," in *ICLR* 2021 Workshop on Geometrical and Topological Representation Learning, 2021.
- [24] M. Jin, Y. Zheng, Y.-F. Li, C. Gong, C. Zhou, and S. Pan, "Multi-scale contrastive siamese networks for self-supervised graph representation learning," arXiv preprint arXiv:2105.05682, 2021.
- [25] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," arXiv preprint arXiv:2006.04131, 2020.
- [26] —, "Graph contrastive learning with adaptive augmentation," in *Proceedings of the Web Conference 2021*, 2021, pp. 2069–2080.
- [27] H. Zhao, X. Yang, Z. Wang, E. Yang, and C. Deng, "Graph debiased contrastive learning with joint representation clustering." in *IJCAI*, 2021, pp. 3434–3440.
- [28] G. Chu, X. Wang, C. Shi, and X. Jiang, "Cuco: Graph representation with curriculum contrastive learning." in *IJCAI*, 2021, pp. 2300–2306.
 [29] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi, "Mixup for node
- [29] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi, "Mixup for node and graph classification," in *Proceedings of the Web Conference 2021*, 2021, pp. 3663–3674.
- [30] V. Verma, M. Qu, K. Kawaguchi, A. Lamb, Y. Bengio, J. Kannala, and J. Tang, "Graphmix: Improved training of gnns for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelli*gence, vol. 35, no. 11, 2021, pp. 10024–10032.
- [31] Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus, "Hard negative mixing for contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21798–21809, 2020.
- [32] S. Zhang, M. Liu, J. Yan, H. Zhang, L. Huang, X. Yang, and P. Lu, "Mmix: Generating hard negatives via multi-sample mixing for contrastive learning," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2461–2470.
- [33] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," arXiv preprint arXiv:1807.03748, 2018.
- [34] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.
- [35] T. N. Kipf and M. Welling, "Variational graph auto-encoders," arXiv preprint arXiv:1611.07308, 2016.
- [36] Z. Liu, H. Feng, and C. Wang, "Rethinking temperature in graph contrastive learning," 2021.