m-MIX: GENERATING HARD NEGATIVES VIA MULTIPLE SAMPLES MIXING FOR CONTRASTIVE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Negative pairs are essential in contrastive learning, which plays the role of avoiding degenerate solutions. Hard negatives can improve the representation ability on the basis of common negatives. Inspired by recent hard negative mining methods via mixup operation in vision, we propose m-mix, which generates hard negatives dynamically. Compared with previous methods, m-mix mainly has three advantages: 1) adaptively chooses samples to mix; 2) simultaneously mixes multiple samples; 3) automatically and comprehensively assigns different mixing weights to the selected mixing samples. We evaluate our method on two image classification datasets, five node classification datasets (PPI, DBLP, Pubmed, etc), five graph classification datasets (IMDB, PTC_MR, etc), and downstream combinatorial tasks (graph edit distance and clustering). Results show that our method achieves state-of-the-art performance on most benchmarks under self-supervised settings.

1 INTRODUCTION

Graph neural networks (GNNs) (Li et al., 2015; Gilmer et al., 2017; Wang et al., 2017) reconcile the expressive power of graphs in modeling interaction with the unparalleled capacity of deep models in learning representations. They process variable-size permutation-invariant graphs and learn low-dimensional representations through an iterative process of transferring, transforming, and aggregating the representations from topological neighbors. However, vanilla GNN training (Kipf & Welling, 2016) calculates loss only from the costly and limited labeled nodes, ignoring the information contained in large amounts of unlabeled nodes. To address this issue, recent studies (Hassani & Khasahmadi, 2020; Zhu et al., 2020; You et al., 2020; Wan et al., 2020; Chen et al., 2020a) introduce contrastive learning (He et al., 2020; Chen et al., 2020b; Oord et al., 2018; Chen et al., 2020c; Chen & He, 2020) into self-supervised learning on graphs. In general, graph-level contrastive methods (Chen et al., 2020b) require a large number of negative samples to avoid degenerate solutions and boost the performance. However, such a large number of negatives are computational and hard to store. Fortunately, mining or generating hard negatives is an efficient way to reduce the number of negatives and improve accuracy, which is important for contrastive learning.

Existing hard negative mining methods are almost from vision, and they can be generally divided into two categories: **(1)** Adversarial based methods (Hu et al., 2020) and **(2)** Mixing based methods (Kim et al., 2020; Lee et al., 2020; Verma et al., 2021; Kalantidis et al., 2020b). Adversarial-based methods update negative samples before updating encoder networks (He et al., 2016), i.e., maximize the similarity between negative pairs before updating encoders, and such a strategy is basically inspired by adversarial training (Goodfellow et al., 2014). Mixing-based methods are inspired by classical data augmentation methods namely mixup (Zhang et al., 2017), which is used to help classify samples close to the boundary. In contrastive learning, mixing based methods generate hard negatives by mixing the positive sample and negative samples with pre-defined mixing weights, which can mainly lead to two problems: **1)** due to sampling the mixed negative sample randomly (Kim et al., 2020) and using static pre-defined mixing weight, the information of similarity between two samples will be ignored (two samples with larger similarity should be sampled with higher probability and mixed with larger weights). **2)** Mix operation is conducted between every two samples, limiting the generated negatives' difficulty for contrastive learning. To address the above two issues, we propose m-mix, which generates hard negatives via multiple samples mixing with different weights.

The main contributions of this paper can be summarized as follows.

1) We propose m-mix to mine hard negatives, which mixes multiple samples and assigns different mixing weights dynamically. Then, we give a theoretical analysis of why the proposed strategy of assigning mixing weights can help generate more difficult hard negatives rather than random. We further give the Rademacher complexity bound (Mohri & Rostamizadeh, 2009) of m-mix. To our best knowledge, this is the first attempt to mix multiple samples in contrastive learning.

2) To emphasize the mixing weights between similar samples (correspondingly, weights of the instance itself will decrease), we propose a diversity loss, greatly improving the stability of m-mix.

3) We further design two revised modules for denoising and better generality, named m-mix-wp and m-mix-op, where the former utilizes the structural information (adjacency matrix) for increasing accuracy in graphs. The latter does not require structural information, which can be applied in vision.

4) For graph, m-mix discards the non-linear projection head (Chen et al., 2020b) and directly contrasts in output space. We theoretically analyze the effect of combinations of sample size and contrastive dimension. Extensive experimental results on both vision and graph data show that the proposed m-mix outperforms most of the previous state-of-the-art methods in self-supervised learning.

2 Related Works

This paper explores hard negatives mining by mixing multiple samples, especially in the context of contrastive learning. We discuss recent contrastive learning and hard negatives mining methods here.

Contrastive learning in Vision. Contrastive learning has become a popular paradigm for self-supervised representation learning on various kinds of data. It works by discriminating positive pairs (two views of the same input image) from negative ones (views of different images). As a pioneering work, CPC (Oord et al., 2018) proposes InfoNCE loss to discriminate positive pairs and negative pairs from sequential data. CMC (Tian et al., 2019) generalizes CPC to multi-view settings, and DIM (Hjelm et al., 2019) introduces information theory interpretation of contrastive learning through local-global mutual information maximization. Later works mainly generate two views of the same input through random, multiple-stage data augmentations like flipping, cropping, resizing, rotation, etc (Hjelm et al., 2019; Chen et al., 2020b). To address the issues of storing a large number of negative samples, MoCo (He et al., 2020) adopts the memory bank (Wu et al., 2018) strategy and uses a momentum-based technique to update two encoders asynchronously. SimCLR (Chen et al., 2020b) directly regards other samples within the same training batch as negatives. Recent works have been paying attention to negative-sample free methods with asymmetric structures (Grill et al., 2020; Chen & He, 2020), or hard negatives (Hu et al., 2020; Robinson et al., 2020; Kalantidis et al., 2020a).

Hard negative mining in contrastive learning. Hard negatives mining refers to generate negative pairs, which are difficult to discriminate. We divide previous hard negatives mining methods into two categories. **1)** Gradient-based. Inspired by adversarial training (Goodfellow et al., 2014), Adco(Hu et al., 2020) tries generating hard negatives via adversarial optimization. In detail, the negatives in the memory bank (He et al., 2020) are first optimized by maximizing contrastive loss, then the encoder network (He et al., 2016) are optimized by minimizing contrastive loss, where the two steps are running alternately. **2)** Mixing based. Inspired by classical data augmentation method Mixup (Zhang et al., 2017) and its numerous variants (Shen et al., 2020; Verma et al., 2019), MoCHi (Kalantidis et al., 2020b) proposes mixing negative samples and positive samples in feature space, where for each positive embedding, MoCHi first finds the most similar negative sample and mixes the two samples' embeddings. Then, i-mix (Lee et al., 2020) proposes mixing two samples in input space, where two samples are first mixed before feeding to the encoder. The concurrent work DACL (Verma et al., 2021) uses the same idea. Further, DACL conducts experiments on both graph and image datasets.

Contrastive learning in Graph. Early works including DGI (Velickovic et al., 2018) and InfoGraph (Sun et al., 2019) adopt the idea of local-global contrastive objective (Hjelm et al., 2019) to node/graph representation learning respectively by contrasting node-graph pairs. Then, MVGRL (Hassani & Khasahmadi, 2020) uses fixed diffusion operations such as heat kernel (Kondor & Lafferty, 2002) and Personalized PageRank (Page et al., 1999) to generate views of the original graph. Then, local-global contrastive objective is adopted and MVGRL achieves state-of-the-art performance on both node classification and graph classification tasks. Inspired by MoCo, GCC (Qiu et al., 2020) generates node views through sub-graph sampling with random walks, where the different sub-graphs are taken as negatives. Then, GRACE (Zhu et al., 2020) and its variant GCA (Zhu et al., 2021)

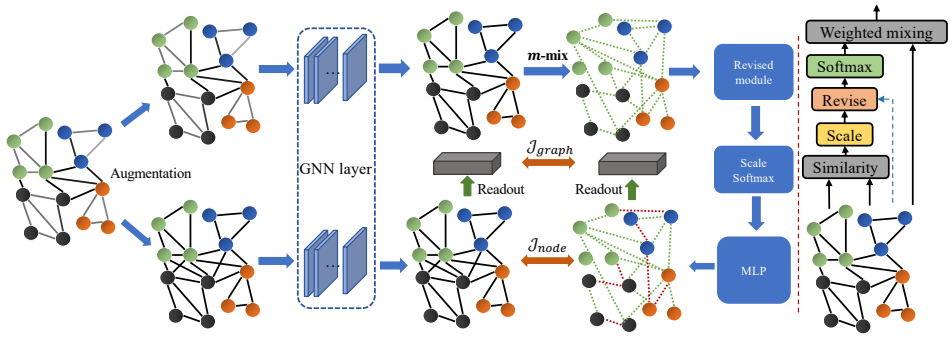


Figure 1: Framework (left) of the proposed m-mix, where the revised module is discussed in "bridge mixing with prior knowledge" and we provide two different revised modules. Green and red dash lines in left figure denote mixing weights $\lambda_i \neq 0$ and $\lambda_i = 0$, respectively. The blue dash line in the right figure means that only m-mix-wp uses the prior knowledge while m-mix-op doesn't. For graph classification, we additionally add graph level contrastive objective function in Eq. 9.

introduce SimCLR to graph, and different nodes are taken as negatives. Depart the success of hard negatives mining in vision, hard negatives in graph contrastive learning is still never explored.

3 The Proposed m-mix

In this section, we present m-mix in detail, starting with preliminaries, followed by the model framework as well as two designed revised modules (m-mix-op for vision and m-mix-wp for graph).

3.1 Preliminaries

Graph neural network. Denote $\mathcal{G} = \{\mathcal{V}, \xi\}$ as a graph, where $\mathcal{V} = \{v_1, v_2, \cdots v_N\}$ and $\xi \in \mathcal{V} \times \mathcal{V}$ represent the node set and the edge set, respectively. Let $\mathbf{X} \in \mathbb{R}^{N \times F}$ be the feature matrix and $\mathbf{A} \in \mathbb{R}^{N \times N}$ be the adjacency matrix, where $\mathbf{x}_i \in \mathbb{R}^F$ is the node feature of v_i and $\mathbf{A}_{ij} = 1$ if edge $(v_i, v_j) \in \xi$. For self-supervised learning without node labeling in $\mathcal{G}$, it aims to learn a GNN encoder $g_\theta(\mathbf{X}, \mathbf{A})$, which takes graph features and adjacency matrix as input, and outputs node/graph semantic embeddings. Generally, GNN learns node representations by aggregating the features of their neighborhood nodes. Formally, we define the l-th layer of GNN is:

$$\mathbf{z}_i^{(l)} = COM^{(l)}\left(\mathbf{z}_i^{(l-1)}, AGG^{(l)}\left(\left\{(\mathbf{z}_i^{(l-1)}, \mathbf{z}_j^{(l-1)}) : j \in \mathcal{N}(i)\right\}\right)\right) \tag{1}$$

where $\mathbf{z}_i^{(l)}$ is the hidden representation of node i at the l-th layer, and $\mathbf{z}_i^{(0)} = \mathbf{x}_i$. $COM(\cdot)$ and $AGG(\cdot)$ are COMBINE and AGGREGATE functions respectively. $\mathcal{N}(i)$ represents the neighborhoods of node v_i. Note that the information of $\mathcal{N}(i)$ is included in adjacency matrix $\mathbf{A}$. On graph-level tasks, a READOUT function will be adopted to summarize all the nodes' representations.

Mixup. There are mainly two kinds of mixup: Geometric-Mixup (Zhou et al., 2021) and Binary-Mixup(Zhang et al., 2017), where the former creates a new sample corresponding to sample $\mathbf{x}$ by taking its weighted-geometric mean with another randomly chosen sample $\widetilde{\mathbf{x}}$. Then the new sample is created by $\mathbf{x}^+ = \mathbf{x}^\lambda \widetilde{\mathbf{x}}^{1-\lambda}$, where λ is the pre-defined value. The later one creates new sample by $\mathbf{x}^+ = \mathbf{x}^\lambda \odot \mathbf{m} + \widetilde{\mathbf{x}} \odot (1 - \mathbf{m})$, where $\mathbf{m}$ means a binary mask from Bernoulli distribution. In line with the previous method (Kalantidis et al., 2020b), we mainly discuss the later kind of mixing.

3.2 Towards Hard Negatives via Multiple Samples Mixing

Multiple samples mixing. Given a set of node features $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$. Through a GNN encoder, we can get the embeddings $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_N\}$, where $\mathbf{Z} \in \mathbb{R}^{N \times \mathcal{F}'}$. N and $\mathcal{F}'$ mean samples size and feature dimension, respectively. Then, for multiple samples mixing, we define a set of mixing weights $\lambda = \{\lambda_i\}_{i=1}^N$, the generated new sample $\hat{\mathbf{z}}_i$ can be formulated as:

$$\hat{\mathbf{z}}_i = \lambda_1 \mathbf{z}_1 + \lambda_2 \mathbf{z}_2 + \cdots + \lambda_N \mathbf{z}_N = \sum_j^N \lambda_j \mathbf{z}_j, \quad s.t. \sum_i^N = 1 \tag{2}$$

In the previous methods (Zhang et al., 2017; Kalantidis et al., 2020b), λ can be statically pre-defined or randomly sampled from Bernoulli distribution, which we argue may not be able to effectively mine the hard negatives completely (we also set the baseline under this condition in the ablation study).

Proposition 1 *Given two negative pairs $(\mathbf{z}_i, \mathbf{z}_j)$ and $(\mathbf{z}_i, \mathbf{z}_k)$, which satisfies $\mathcal{H}(\mathbf{z}_i, \mathbf{z}_j) > \mathcal{H}(\mathbf{z}_i, \mathbf{z}_k)$ and $\mathcal{H}(\cdot, \cdot)$ is a similarity metric function, assign larger mixing weight to pair $(\mathbf{z}_i, \mathbf{z}_j)$ than $(\mathbf{z}_i, \mathbf{z}_k)$ will generate more difficult negative pair.*

Quantification of mixing weights. The proof is given in Appendix C. Proposition 1 says similar negatives pairs should be assigned larger mixing weights. Thus, to quantify the mixing weights, we design two methods: one to learn mixing weights in original space and the other is in space after linear projection. For the former case, we can quantify the mixing weights in each iteration by:

$$\lambda_j = \frac{\exp(\mathcal{H}(\mathbf{z}_i, \mathbf{z}_k))}{\sum_k \exp(\mathcal{H}(\mathbf{z}_i, \mathbf{z}_k))} \quad s.t. \quad 0 <= j <= N \tag{3}$$

where $\mathcal{H}$ means a similarity metric function. In this paper, we use dot product after l_2 normalization. This definition is simple yet shown empirically effective in our experiments. We denote the above method as mo-mix. However, the similarity is often measured in semantic space rather than in the original space. Besides, the definition in Eq. 3 ignores homogenization (Vaswani et al., 2017), i.e., although two instances $\mathbf{z}_i$ and $\mathbf{z}_j$ are similar in original space, but the mixing weights of $\mathbf{z}_i -> \mathbf{z}_j$ and $\mathbf{z}_j -> \mathbf{z}_i$ may should be different in many cases (Kipf & Welling, 2016; Veličković et al., 2017). Hence, we propose another approach to quantify mixing weights in two different spaces as follow:

$$\lambda_j = \frac{\exp(\mathcal{H}(\mathbf{z}_i^\top \mathbf{p}_m, \mathbf{z}_j^\top \mathbf{p}_n))}{\sum_k \exp(\mathcal{H}(\mathbf{z}_i^\top \mathbf{p}_m, \mathbf{z}_k^\top \mathbf{p}_n))} \quad s.t. \quad 0 <= j <= N \tag{4}$$

where $\mathbf{p}_m$ and $\mathbf{p}_n$ are two learnable weights. By the two learnable weights with different values, the above issue can be mitigated. We denote the definition in Eq. 4 as mp-mix.

Reduce self-mixing weights. Recall we aim to increase the mixing weights between similar negative pairs. However, mixing weight of the sample itself is also an important parameter. Take an example, for $\mathbf{z}_i$ in Eq. 2, if λ_i is very large (correspondingly, different λ_j will be very small), the hard negatives may not be generated. To address this issue, we pertinently propose two methods targeted to mo-mix and mp-mix. For mo-mix, it directly computes the similarity of original space, so we directly set the $\lambda_i = C$, where C is a pre-defined value. Correspondingly, mixing weights of other instances are:

$$\lambda_j = \frac{(1 - C)\exp(\mathcal{H}(\mathbf{z}_i, \mathbf{z}_j)}{\sum_k \exp(\mathcal{H}(\mathbf{z}_i, \mathbf{z}_k)} \quad s.t. \quad 0 <= j <= N, \ j \neq i \tag{5}$$

While mp-mix is more flexible than mo-mix, as mp-mix has two learnable weights to trade-off. We design a new diversity loss to reduce the mixing weights of the instance itself as follow:

$$\mathcal{L}_{Div} = -\frac{1}{N}\sum_i \left(\frac{\mathbf{z}_i^\top \mathbf{p}_m}{\|\mathbf{z}_i^\top \mathbf{p}_m\|_2} - \frac{\mathbf{z}_i^\top \mathbf{p}_n}{\|\mathbf{z}_i^\top \mathbf{p}_n\|_2} \right)^2 = 2 \cdot \frac{(\mathbf{z}_i^\top \mathbf{p}_m)^\top (\mathbf{z}_i^\top \mathbf{p}_n)}{\|\mathbf{z}_i^\top \mathbf{p}_m\|_2 \|\mathbf{z}_i^\top \mathbf{p}_n\|_2} - 2 \tag{6}$$

By the diversity loss, we project features into two diversified spaces, reducing self-mixing weights.

Bridge mixing with prior knowledge. Although mo-mix and mp-mix are comprehensive enough, there are still two practical issues to consider. First, for each sample, we will compute the similarity in the original space or projected space of $N - 1$ other instances, which is computational and expensive; Second, each generated new sample is mixed by all the old instances, which may bring much noise. To address these issues, we further design two revised modules. 1) Utilize the prior knowledge of the adjacency matrix. Usually, nodes and its' neighbors have some similar properties, which makes them more similar than other nodes. Thus, one of the practical solutions is for each node, narrowing down the mining scope of mixing to corresponding neighbor nodes and we denote this method m-mix-wp (with prior). 2) Pre-define a threshold θ, and if $\mathcal{H}(\mathbf{z}_i, \mathbf{z}_j) < \theta$ or $\mathcal{H}(\mathbf{z}_i^\top \mathbf{p}_m, \mathbf{z}_j^\top \mathbf{p}_n) < \theta$, we set $\lambda_j = 0$ and we denote this method m-mix-op (without prior). Although both two revised modules can solve the mentioned problems, m-mix-wp achieves better performance than m-mix-op (due to prior knowledge). However, In the vision domain, there is no prior knowledge of the adjacency matrix. Thus, we can only use the m-mix-op for image data. We denote m-mix-wp as m-mix later.

Relation to GAT. Our m-mix is somewhat similar to GAT (Veličković et al., 2017), since both m-mix and GAT use similarity to re-weights. We clarify the differences between them here. **1)**

Methodology, GAT requires the structural information (adjacency matrix), while m-mix-op doesn't. This also makes m-mix-op can be applied in vision and other domains, while GAT can't. Besides, the diversified loss is proposed to reduce the mixing weights of the node itself. **2) Technology**, GAT uses concatenate operation and a learnable weight to calculate similarity, which aims to increase the representation ability of GAT, while both mp-mix-op and mp-mix-wp directly adopt normalized dot product in original space or projected space, which aims to generate more difficult negative pairs. We also give **experimental** comparison between GAT and mp-mix in Table 4.

3.3 Framework and Objective

For graph $\mathbf{G} = (\mathbf{X}, \mathbf{A})$, we first generate two views $\mathbf{G}^a = (\mathbf{X}^a, \mathbf{A}^a)$, $\mathbf{G}^b = (\mathbf{X}^b, \mathbf{A}^b)$ by graph augmentation. Then, take the generated views to GNN encoder f, we can obtain the node representations of two graph views $\mathbf{Z}^a$, $\mathbf{Z}^b$. Then, for branch $\mathbf{Z}^a$, we generate hard negatives $\hat{\mathbf{Z}}^a$ by the proposed m-mix followed an MLP module in graph datasets. For node i in view a, the node-level loss is:

$$\mathcal{L}_{info-N}(g(\hat{\mathbf{z}}_i^a)) = -\log \frac{e^{\mathcal{H}(g(\hat{\mathbf{z}}_i^a), \mathbf{z}_i^b)/\tau}}{\sum_{j=1, j\neq i}^{N} e^{\mathcal{H}(g(\hat{\mathbf{z}}_i^a), g(\hat{\mathbf{z}}_j^a))/\tau} + \sum_{j=1}^{N} e^{\mathcal{H}(g(\hat{\mathbf{z}}_i^a), \mathbf{z}_j^b)/\tau}}, \tag{7}$$

where g denotes the MLP module, which is commonly used in previous methods (Grill et al., 2020; Chen & He, 2020). The objective is similar to view b. $\mathcal{H}(\cdot, \cdot)$ means similarity metric function between two vectors (should be the same with $\mathcal{H}$ in Eq. 3 and Eq. 4), and τ means temperature hyper-parameter (Chen et al., 2020b). Then, the total objectives can be formulated as:

$$\mathcal{J}_{node}(\mathcal{G}) = \frac{1}{2N} \sum_{u} \left[\mathcal{L}_{info-N}(\hat{\mathbf{z}}_i^a) + \mathcal{L}_{info-N}(\mathbf{z}_i^b)\right] + \mathcal{L}_{div} \tag{8}$$

The above objective can capture local information (node-level), while for graph level tasks (graph classification, graph edit distance), we need global information (graph-lvel). Hence, we design graph-level contrastive objectives. Consider given M graphs, and denote the graph representation matrix as $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_M\}$, where $\mathbf{H} \in \mathbb{R}^{M \times \mathcal{F}'}$. The objectives can be formulated as:

$$\mathcal{J}_{graph} = -\frac{1}{M} \left[\sum_{i=1}^{M} \log \frac{e^{\mathcal{H}(\mathbf{h}_i^a, \mathbf{h}_i^b)}}{\sum_{j, j\neq i} e^{\mathcal{H}(\mathbf{h}_i^a, \mathbf{h}_j^a)} + \sum_j e^{sim(\mathbf{h}_i^a, \mathbf{h}_j^b)}} + \mathcal{J}_{node}(\mathcal{G}_i) \right] \tag{9}$$

3.4 Theoretical Analysis

Theorem 1 *(Empirical Rademacher Complexity of m-mix.) Given a hypothesis set $\mathcal{H}_f$ and a set of samples $\mathcal{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_N\}$, we assume that $\mathfrak{R}_{\mathcal{Z}}(\mathcal{H}_f)$ and $\mathfrak{R}^m_{\mathcal{Z}}(\mathcal{H}_f)$ are the empirical Rademacher complexity of hypothesis class $\mathcal{H}_f$ of original data and after m-mix mixed data. Then, the different between two Redemacher complexity $\mathfrak{R}_{\mathcal{Z}}(\mathcal{H}_f) - \mathfrak{R}^m_{\mathcal{Z}}(\mathcal{H}_f)$ is less than or equal to a constant multiple of the sample variance of the norm of the input samples:*

$$\mathfrak{R}_{\mathcal{Z}}(\mathcal{H}_f) - \mathfrak{R}^m_{\mathcal{Z}}(\mathcal{H}_f) \leq C\sqrt{\frac{s^2 \|x\|_2}{N}} \tag{10}$$

where C is a constant related to self mixing weight and s^2 is the sample variance of sample set.

The above result gives Rademacher complexity of m-mix data, where the proof are given in Appendix C. Besides, since $\mathfrak{R}^m_{\mathcal{Z}}(\mathcal{H}_f) \leq \mathfrak{R}_{\mathcal{Z}}(\mathcal{H}_f)$, data generated by m-mix allows the neural networks more robust (also influenced by sample size N and self mixing weight λ_i).

Theorem 2 *(Suitable contrastive dimension without projection head). Denote $\mathcal{Y} \in \mathbb{R}^{N \times \mathcal{U}}$ as the one-hot label matrix, where $\mathcal{U}$ is the number of class. Assume the contrastive dimension is large enough i.e., $\mathcal{F}' > N$. When $\mathcal{U} << N < \mathcal{F}'$ which mostly holds in practice, then for any $\delta > 0$ and $C > 0$, if the sample size satisfies:*

$$N > \frac{\mu^2}{C} \left(\log \left(7 + \frac{1}{N} \right) + 2 \log \mathcal{F}' - \log \delta \right), \tag{11}$$

then with probability at least $1 - \delta$, $\Phi = \mathcal{W}Z - 2\tau^{-1} \|\mathcal{W}\epsilon\|_\infty I_{\mathcal{F}'}$ is restricted diagonally dominant with sparsity s, where $\mathcal{W}$ is the least-square projection $Z^\top (ZZ^\top)$, and ϵ is the learned noise in contrastive dimension.

Table 1: Mean accuracy for supervised and unsupervised learning on node classification. $\mathbf{X}$: node features; $\mathbf{A}$: adjacency matrix; $\mathbf{Y}$: labels; $\mathbf{D}$: diffusion matrix in (Tian et al., 2019); $\mathbf{S}$: affinity matrix in this paper; †: our reproduce otherwise the numbers are quoted from original papers by default. Diff means we use diffusion as data augmentation.

	method \ dataset	input mode	Cora	Citeseer	Pubmed
supervised	MLP (Velickovic et al., 2018)	$\mathbf{X}$, $\mathbf{Y}$	55.1	46.5	71.4
	ICA (Getoor, 2005)	$\mathbf{A}$,$\mathbf{Y}$	75.1	69.1	73.9
	LP (Zhu et al., 2003)	$\mathbf{A}$,$\mathbf{Y}$	68.0	45.3	63.0
	MANIREG (Belkin et al., 2006)	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$	59.5	60.1	70.7
	SEMIEMB (Weston et al., 2012)	$\mathbf{X}$, $\mathbf{Y}$	59.0	59.6	71.7
	PLANETOID (Yang et al., 2016)	$\mathbf{X}$, $\mathbf{Y}$	75.7	64.7	77.2
	CHEBISHEV (Defferrard et al., 2016)	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$	81.2	69.8	74.4
	GCN (Kipf & Welling, 2016)	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$	81.5	70.3	79.0
	MONET (Monti et al., 2017)	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$	81.7 ± 0.5	–	78.8 ± 0.3
	JKNET (Xu et al., 2018b)	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$	82.7 ± 0.4	$\mathbf{73.0 \pm 0.5}$	77.9 ± 0.4
	GAT (Velickovic et al., 2018)	$\mathbf{X}$, $\mathbf{A}$, $\mathbf{Y}$	$\mathbf{83.0 \pm 0.7}$	72.5 ± 0.7	$\mathbf{79.0 \pm 0.3}$
unsupervised	LINEAR (Velickovic et al., 2018)	$\mathbf{X}$	47.9 ± 0.4	49.3 ± 0.2	69.1 ± 0.3
	DEEPWALK (Perozzi et al., 2014)	$\mathbf{X}$,$\mathbf{A}$	70.7 ± 0.6	51.4 ± 0.5	74.3 ± 0.9
	GAE (Kipf & Welling, 2016)	$\mathbf{X}$,$\mathbf{A}$	71.5 ± 0.4	65.8 ± 0.4	72.1 ± 0.5
	VERSE (Tsitsulin et al., 2018)	$\mathbf{X}$, $\mathbf{D}$, $\mathbf{A}$	72.5 ± 0.3	55.5 ± 0.4	–
	DGI (Velickovic et al., 2019)	$\mathbf{X}$, $\mathbf{A}$	82.3 ± 0.6	71.8 ± 0.7	76.8 ± 0.6
	DGI† (Velickovic et al., 2019)	$\mathbf{X}$, $\mathbf{D}$	83.5 ± 0.7	71.8 ± 0.4	78.0 ± 0.4
	GRAPHCL (You et al., 2020)	$\mathbf{X}$, $\mathbf{A}$	82.3 ± 0.1	73.1 ± 0.2	–
	GRAPHCL† (You et al., 2020)	$\mathbf{X}$, $\mathbf{A}$	83.4 ± 0.7	72.4 ± 0.4	79.4 ± 0.7
	GRACE† (Zhu et al., 2020)	$\mathbf{X}$, $\mathbf{A}$	82.2 ± 0.6	71.6 ± 0.6	78.4 ± 0.5
	MVGRL (Hassani & Khasahmadi, 2020)	$\mathbf{X}$, $\mathbf{D}$, $\mathbf{A}$	$\mathbf{86.8 \pm 0.5}$	73.3 ± 0.5	80.1 ± 0.7
	MVGRL† (Hassani & Khasahmadi, 2020)	$\mathbf{X}$, $\mathbf{D}$, $\mathbf{A}$	84.1 ± 0.9	73.5 ± 0.6	80.4 ± 1.1
ours	MVGRL+(mo-mix)	$\mathbf{X}$, $\mathbf{D}$, $\mathbf{A}$	85.4 ± 0.4	75.1 ± 0.4	82.6 ± 0.2
	MVGRL+(mp-mix w/o div)	$\mathbf{X}$, $\mathbf{D}$, $\mathbf{A}$	85.1 ± 0.8	74.4 ± 0.9	81.8 ± 0.7
	MVGRL+(mp-mix)	$\mathbf{X}$, $\mathbf{D}$, $\mathbf{A}$	85.9 ± 0.3	$\mathbf{75.8 \pm 0.3}$	$\mathbf{82.9 \pm 0.3}$

This property implies that Φ is actually screening consistent variable for any $\beta \in \mathcal{B}_\tau(s,\rho)$, where $\mathcal{B}_\tau(s,\rho) = \{\beta \in \mathcal{R}^{\mathcal{F}'} : \min_{i \in supp(\beta)} |\beta_i| \geq \tau, supp(\beta) \leq s, \frac{\max_{i \in supp(\beta)} |\beta_i|}{\min_{i \in supp(\beta)} |\beta_i|} \leq \rho\}$ and $supp(f)$ means support set of f. Note that the set of β is the obtained least square solution, i.e., $\beta = \mathcal{W}\mathcal{Y}$, where $\mathcal{Y}$ is the label. The proof is given in Appendix C. We analyze on both optimal solution and non-optimal solution for contrastive learning. We can draw a conclusion for least square solver, and the sample size and contrastive dimension should be appropriate (note that not the larger the better). This is consistent with our experimental results (see Fig. 2 and Fig. 3).

4 Experiments

4.1 Experimental setups

Dataset and implementation. For graph datasets, we consider five popular node classification benchmarks: Cora, Citeseer, Pubmed, DBLP and PPI, and five graph classification benchmarks: MUTAG (Kriege & Mutzel, 2012), PTC_MR (Kriege & Mutzel, 2012), IMDB-BINARY and IMDB-MULTI (Yanardag & Vishwanathan, 2015), REDDIT-BINARY (Yanardag & Vishwanathan, 2015). For data augmentation, we consider mo-mix and mp-mix can be applied in methods with simclr-like framework, hence, we choose edge dropping, node feature masking in GRACE (Zhu et al., 2020) and diffusion in MVGRL (Hassani & Khasahmadi, 2020). The model is implemented with PyTorch and each trial is executed on a single Tesla V100 GPU. For image datasets, we main evaluate our method on basis of SimCLR on CIFAR-10 and CIFAR-100 benchmarks, where the data augmentation in images is in line with SimCLR (Chen et al., 2020b), i.e., Random Resized Crop to 32×32, Random Horizontal Flip, Color Jitter and Gray Scale. See Appendix A for more details of hyper-parameters.

4.2 Comparison with previous arts

Baselines. We compare our method with both supervised and unsupervised learning methods. On node classification tasks, we compare our method on recent popular self-supervised methods, such as GRAPH-CL (You et al., 2020), MVGRL (Hassani & Khasahmadi, 2020). Results show our

Table 2: Mean 10-fold cross validation accuracy on supervised and unsupervised learning. RANDOM: Random walk; N2VEC, S2VEC, G2VEC: Node to vector, sub-graph to vector, graph to vector; w/o div: our method without diversity loss in Eq. 6

	methods \ datasets	MUTAG	PTC_MR	IMDB-BIN	IMDB-MULTI	REDDIT-BIN
kernel	SP (Borgwardt & Kriegel, 2005)	85.2 ± 2.4	58.2 ± 2.4	55.6 ± 0.2	38.0 ± 0.3	64.1 ± 0.1
	GK (Shervashidze et al., 2009)	81.7 ± 2.1	57.3 ± 1.4	65.9 ± 1.0	43.9 ± 0.4	77.3 ± 0.2
	WL (Shervashidze et al., 2011)	80.7 ± 3.0	58.0 ± 0.5	**72.3 ± 3.4**	**47.0 ± 0.5**	68.8 ± 0.4
	DGK (Yanardag & Vishwanathan, 2015)	87.4 ± 2.7	60.1 ± 2.6	67.0 ± 0.6	44.6 ± 0.5	**78.0 ± 0.4**
	MLG (Kondor & Pan, 2016)	**87.9 ± 1.6**	**63.3 ± 1.5**	66.6 ± 0.3	41.2 ± 0.0	–
supervised	GRAPHSAGE (Hamilton et al., 2017)	85.1 ± 7.6	63.9 ± 7.7	72.3 ± 5.3	50.9 ± 2.2	–
	GCN (Kipf & Welling, 2016)	85.6 ± 5.8	64.2 ± 4.3	74.0 ± 3.4	51.9 ± 3.8	50.0 ± 0.0
	GIN-0 (Xu et al., 2018a)	89.4 ± 5.6	64.6 ± 7.0	**75.1 ± 5.1**	**52.3 ± 2.8**	**92.4 ± 2.5**
	GIN-ϵ (Xu et al., 2018a)	89.0 ± 6.0	63.7 ± 8.2	74.3 ± 5.1	52.1 ± 3.6	92.2 ± 2.3
	GAT-ϵ (Velickovic et al., 2018)	89.4 ± 6.1	**66.7 ± 5.1**	70.5 ± 2.3	47.8 ± 3.1	85.2 ± 3.3
	PATCHY (Niepert et al., 2016)	**92.6 ± 4.2**	60.0 ± 4.8	71.0 ± 2.2	45.2 ± 2.8	86.3 ± 1.6
unsupervised	RANDOM (Gärtner et al., 2003)	83.7 ± 1.5	57.9 ± 1.3	50.7 ± 0.3	34.7 ± 0.2	–
	N2VEC (Grover & Leskovec, 2016)	72.6 ± 10.2	58.6 ± 8.0	–	–	–
	S2VEC (Adhikari et al., 2018)	61.1 ± 15.8	60.0 ± 6.4	55.3 ± 1.5	36.7 ± 0.8	71.5 ± 0.4
	G2VEC (Narayanan et al., 2017)	83.2 ± 9.6	60.2 ± 6.9	71.1 ± 0.5	50.4 ± 0.9	75.8 ± 1.0
	INFOGRAPH (Sun et al., 2019)	89.0 ± 1.1	61.7 ± 1.4	73.0 ± 0.9	49.7 ± 0.5	82.5 ± 1.4
	GRAPHCL (You et al., 2020)	86.8 ± 1.3	–	71.1 ± 0.4	–	**89.5 ± 0.8**
	GCC-MOCO (Qiu et al., 2020)	–	–	73.8 ± 1.1	50.3 ± 0.8	87.6 ± 0.9
	GCC-RAND (Qiu et al., 2020)	–	–	**75.6 ± 0.9**	50.9 ± 0.6	87.8 ± 0.7
	MVGRL (Hassani & Khasahmadi, 2020)	89.7 ± 1.1	62.5 ± 1.7	74.2 ± 0.7	51.2 ± 0.5	84.5 ± 0.6
	MVGRL† (Hassani & Khasahmadi, 2020)	90.1 ± 0.7	62.2 ± 1.1	74.4 ± 0.8	51.2 ± 0.7	85.1 ± 0.4
ours	MVGRL+(mo-mix)	91.8 ± 0.4	64.5 ± 0.4	75.1 ± 0.7	52.6 ± 0.5	88.4 ± 0.4
	MVGRL+(mp-mix w/o div)	90.7 ± 1.3	63.7 ± 0.9	75.1 ± 0.7	52.0 ± 0.7	88.1 ± 0.6
	MVGRL+(mp-mix)	**92.0 ± 0.5**	**64.6 ± 0.4**	**75.6 ± 0.5**	**52.9 ± 0.4**	88.7 ± 0.4

methods (both mo-mix and mp-mix) achieve state-of-the-art performance on Citeseer and Pubmed. On graph classification tasks, we compare our result with GCC (Qiu et al., 2020), MVGRL (Hassani & Khasahmadi, 2020) and GRAPH-CL (You et al., 2020). Experimental results show that our method outperforms all of the unsupervised methods except on REDDIT-BINARY. For the results of other baseline models, we follow the results reported by (Hassani & Khasahmadi, 2020).

Node classification. The results of node classification in Table 1 show that mp-mix achieves state-of-the-art results over both unsupervised and self-supervised methods. Specifically, mp-mix outperforms existing unsupervised methods by 2.3% and 2.5% accuracy on Citeseer and Pubmed. mo-mix outperforms 1.6% and 1.2% accuracy than original baseline MVGLR. Note that our reproduced accuracy of MVGRL on Cora dataset doesn't match with the original paper reported. Other researchers also point out this problem in these sites[1][2]. Furthermore, we also report mp-mix without diversity loss, and the results show diversity loss can make m-mix more stable, also boost accuracy.

Graph classification. Table 2 gives classification accuracy on graph, where MVGRL+(mp-mix) outperforms existing state-of-the-art methods (Hassani & Khasahmadi, 2020; Qiu et al., 2020) by 1.9%, 2.4%, 1.7% on MUTAG, PTC_MR, IMDB-MULTI, respectively. On IMDB-BIN and REDDIT-BIN, mp-mix achieves 1.2% and 3.6% accuracy rather than baseline MVGRL.

Comparison of m-mix-wp, m-mix-op and binary mix. In Section 3, we design an approach to avoid utilizing prior adjacency matrix named m-mix-wp. We evaluate this approach in both vision and graph domains. We also compare our method with binary mixing. For the implementation of binary mixing, we choose the most similar negative sample of each query positive sample in one batch and mix them with mixing weight 0.5 as default. In vision datasets, we choose SimCLR (Chen et al., 2020b) as baselines, and use ResNet-18 (He et al., 2016) as backbone. We set max epoch as 500, and use Adam optimizer with learning rate 1e-3. We evaluate the pre-trained model with a linear model, where the baseline code is from this site[3]. Table 3 shows the accuracy with different mixing strategies, where multiple samples mixing achieves better results rather than binary mixing. Note that there's no prior adjacency matrix information in vision. Thus, we only compare our m-mix-op

[1] https://github.com/kavehhassani/mvgrl/issues/2
[2] https://github.com/hengruizhang98/mvgrl
[3] https://github.com/leftthomas/SimCLR

Table 3: Top-1 accuracy on graph and vision datasets with different mixing strategies.

Method	Cora	Citeseer	PubMed	Method	CIFAR-10	CIFAR-100
MVGRL	84.3 ± 1.1	73.1 ± 0.7	80.1 ± 0.8	SimCLR	89.21	61.53
MVGRL+(binary-mix)	84.9 ± 0.9	74.1 ± 0.5	81.4 ± 0.7	SimCLR+(binary-mix)	89.47	61.82
MVGRL+(mo-mix-op)	85.4 ± 0.4	75.1 ± 0.4	82.6 ± 0.2	SimCLR+(mo-mix-op)	**90.09**	**62.86**
MVGRL+(mo-mix-wp)	**85.9 ± 0.3**	**75.8 ± 0.3**	**82.9 ± 0.3**	SimCLR+(mo-mix-wp)	∼	∼

Table 4: Mean accuracy on DBLP and PPI datasets. GRACE† means our reproduce result. It is modified to GRACE (GAT) by replacing the encoder with GAT.

Dataset	Protocol	GRACE	GRACE†	GRACE (GAT)	GRACE+(mp-mix w/o div)	GARCE+(mp-mix)
DBLP	Transductive	84.2 ± 0.1	83.8 ± 0.2	84.3 ± 0.2	85.2 ± 0.2	**85.6 ± 0.1**
PPI	Inductive	66.2 ± 0.1	66.3 ± 0.2	66.3 ± 0.1	67.3 ± 0.2	**67.8 ± 0.1**

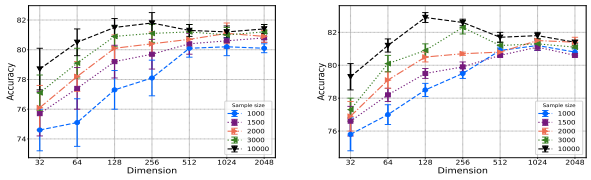
Table 5: Accuracy with different mixing methods across seven graph datasets. The first line MVGRL means taking normalized adjacency and diffusion matrix as input to perform contrasting and the objective function is the same as (Hassani & Khasahmadi, 2020). The remaining lines use objectives of this paper $\mathcal{J}_{node}$ in Eq. 8 and $\mathcal{J}_{graph}$ in Eq. 9. MVGRL+(random mix) means the designed baseline. Here we report the results with best-tuned hyper-parameters under different methods.

task / method	node classification			graph classification			
	Cora	Citeseer	Pubmed	MUTAG	PTC_MR	IMDB-BIN	IMDB-MULTI
MVGRL	84.1 ± 0.9	73.5 ± 0.6	80.4 ± 1.1	90.1 ± 0.7	62.2 ± 1.1	74.4 ± 0.8	51.2 ± 0.7
MVGRL†	84.3 ± 1.1	73.1 ± 0.7	80.1 ± 0.8	89.7 ± 0.8	62.8 ± 1.0	73.8 ± 0.4	51.3 ± 0.7
MVGRL+(random-mix)	84.1 ± 1.6	73.8 ± 1.1	81.1 ± 1.0	89.9 ± 1.3	63.1 ± 1.4	74.2 ± 1.0	52.0 ± 0.9
MVGRL+(mo-mix)	85.4 ± 0.4	75.1 ± 0.4	82.6 ± 0.2	91.8 ± 0.4	64.5 ± 0.4	75.1 ± 0.7	52.6 ± 0.5
MVGRL+(mp-mix)	**85.9 ± 0.3**	**75.8 ± 0.3**	**82.9 ± 0.3**	**92.0 ± 0.5**	**64.6 ± 0.4**	**75.6 ± 0.5**	**52.9 ± 0.4**

with binary-mix. The reason why the accuracy of m-mix-op is slightly lower than m-mix-wp is that m-mix-wp utilizes the prior knowledge while m-mix-op doesn't. We think one of the directions worth exploring is estimating "adjacency knowledge" across images in vision.

4.3 Ablation Study

Comparison with GAT. As discussed in Section 3, mp-mix without diversity loss is somewhat similar to GAT (Veličković et al., 2017), while mo-mix and mp-mix are completely different from GAT on both method and technology. Here, we further design experiments to clarify the difference. We evaluate mp-mix with the baseline GRACE (Zhu et al., 2020) on DBLP and PPI benchmarks. Table 4 provides accuracy. After replacing the backbone GNN in GRACE with GAT, we only get 0.4% and 0.0% accuracy improvement on DBLP and PPI datasets. However, when we use multiple samples mixing methods, mp-mix outperforms baseline GRACE 1.4% and 1.6% accuracy on DBLP and PPI, respectively.



(a) Without diversity loss (b) With diversity loss

Figure 2: Classification accuracy on Pubmed dataset with different sample sizes and different hidden dimensions.

Mixing weights. To further explore the effect of different mixing weights, we design a baseline, i.e., random generate mixing weights from Gaussian distribution and regularize them with Softmax function. Table 5 shows the accuracy of different mixing weights. Although randomly generated value can get a little improvement against baseline MVGRL, it suffers from instability (variance is large). Then, by replacing the random mix with the proposed mo-mix and mp-mix with diversity loss, the performance is improved with a large range in both accuracy and stability.

Contrastive dimension and sample size. Contrastive learning usually requires a large number of negative pairs to learn more information from the negatives. To evaluate the robustness of our method

Table 6: Accuracy (top) and AUC (bottom) score on COIL-DEL. HIST-KERNEL means vertex edge hist kernel; n means number of nodes in sampled sub-graph. We use the implementation of the two kernel methods, i.e., HIST-KERNEL (Hido & Kashima, 2009) and WL-KERNEL (Shervashidze et al., 2011) provided by (Sugiyama et al., 2018) to produce the reported results.

	n	HIST-KERNEL	WL-KERNEL	supervised	MVGRL	MVGRL+(mp-mix)
Accuracy	50	0.153 ± 0.131	$\mathbf{0.934 \pm 0.017}$	0.813 ± 0.019	0.713 ± 0.034	0.741 ± 0.039
	20	0.160 ± 0.103	0.826 ± 0.014	0.783 ± 0.031	0.824 ± 0.021	$\mathbf{0.856 \pm 0.011}$
	15	0.138 ± 0.076	0.791 ± 0.037	0.814 ± 0.014	$\mathbf{0.847 \pm 0.014}$	0.842 ± 0.017
	10	0.129 ± 0.059	0.751 ± 0.034	0.903 ± 0.009	0.891 ± 0.032	$\mathbf{0.911 \pm 0.008}$
AUC	50	0.501 ± 0.001	0.525 ± 0.001	0.587 ± 0.009	0.649 ± 0.014	$\mathbf{0.663 \pm 0.021}$
	20	0.507 ± 0.002	0.564 ± 0.008	0.782 ± 0.009	0.788 ± 0.005	$\mathbf{0.794 \pm 0.003}$
	15	0.503 ± 0.001	0.594 ± 0.017	0.781 ± 0.031	0.801 ± 0.015	$\mathbf{0.813 \pm 0.024}$
	10	0.507 ± 0.003	0.626 ± 0.009	0.832 ± 0.011	$\mathbf{0.861 \pm 0.007}$	0.854 ± 0.010

w.r.t the number of negative samples (i.e., subgraph sample size), we try different combinations of sample size and contrastive dimension. Fig. 2(a) shows the result on Pubmed. Different from the conclusion in vision (He et al., 2020; Chen et al., 2020b), showing that using large contrastive dimension can get better results (saturate at 4096), we find that a proper combination of contrastive dimension and the sample size is important in graph contrastive learning, which is in line with Theorem 2. Empirically, a smaller contrastive dimension (i.e., 256) forces to pull samples in a lower-dimensional hyper-sphere uniformly, which is more difficult than pulling in a large dimensional hyper-sphere. However, when the contrastive dimension is too small (32), the information can not be fully represented. Thus, The best accuracy is obtained when the contrastive dimension is set as 256. On the other hand, we can not ignore the advantage of large contrastive dimension, that is, the stability (standard-deviation, see error bar in Fig. 2) of m-mix can significantly improve and the convergence rate (see Fig. 3 in Appendix B) is faster than that of low contrastive dimension.

Transferability in downstream tasks. To evaluate transferability of m-mix in downstream tasks, we conduct extensive experiments on node clustering and graph edit distance, which follows (Li et al., 2019). We conduct graph edit experiments on (Riesen & Bunke, 2008). In line with (Riesen & Bunke, 2008), our pre-trained models are evaluated by two metrics: 1) pair AUC - the area under the ROC curve for classifying pairs of graphs as similar or not on a fixed set of 1000 pairs and 2) triplet accuracy - the accuracy of correctly assigning higher similarity to the positive pair in a triplet than the negative pair on a fixed set of 1000 triplets. The comparison of edit distance mainly includes two kinds of methods: 1) kernel-based, such as HIST-KERNEL (Hido & Kashima, 2009) and WL-KERNEL (Shervashidze et al., 2011). 2) Deep learning-based, i.e., supervised and MVGRL pre-trained. The results in Table 6 show that our method outperforms other GNN methods. We find that with a larger number of nodes, kernel-based methods outperform GNN-based methods. That's because kernel-based methods treat each node equally, while GNN-based methods are likely to pay more attention to important nodes in a sub-graph. This also explains why the AUC score of GNN-based methods is higher than kernel-based methods but has a lower Accuracy.

5 Conclusion and Discussions

In this paper, we have proposed a new mp-mix to mine hard negatives for contrastive learning, which mixes multiple samples and assigns different mixing weights dynamically. To our best knowledge, this is the first attempt at mixing multiple samples, and also the first work for learning mixing weights dynamically. Then, we analyze why the proposed strategy of assigning mixing weight is better than others. We further provide theoretical analysis to Redamacher complexity bound of mp-mix, and we also provide theoretical analysis on the relation of contrastive dimension and sample size. The experiments are conducted on two image classification datasets, five node classification datasets, and five graph classification datasets, where our method achieves state-of-the-art performance on most of the datasets. To further evaluate our method's transferability, we conduct extensive downstream experiments on clustering and graph edit distance. The results imply our method can bring a stable model with high transferability. Due to the high generality of our method, we hope our method can be applied in other domains, such as NLP (Gao et al., 2021) and cross modal (Conde & Turgutlu, 2021) pretraining, which we leave in future work.

REFERENCES

Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. Sub2vec: Feature learning for subgraphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 170–182. Springer, 2018.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7 (11), 2006.

Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pp. 8–pp. IEEE, 2005.

Deli Chen, Yanyai Lin, Lei Li, Xuancheng Ren Li, Jie Zhou, Xu Sun, et al. Distance-wise graph contrastive learning. *arXiv preprint arXiv:2012.07437*, 2020a.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020b.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020c.

Marcos V Conde and Kerem Turgutlu. Clip-art: Contrastive pre-training for fine-grained art classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3956–3960, 2021.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*, pp. 129–143. Springer, 2003.

Lise Getoor. Link-based classification. In *Advanced methods for knowledge discovery from complex data*, pp. 189–207. Springer, 2005.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272. PMLR, 2017.

Xavier Glorot and Yoshua Bengio. 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.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

Aditya Grover and Jure Leskovec. 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.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

Shohei Hido and Hisashi Kashima. A linear-time graph kernel. In *2009 Ninth IEEE International Conference on Data Mining*, pp. 179–188. IEEE, 2009.

R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

Qianjiang Hu, Xiao Wang, Wei Hu, and Guo-Jun Qi. Adco: Adversarial contrast for efficient learning of unsupervised representations from self-trained negative adversaries. *arXiv preprint arXiv:2011.08435*, 2020.

Yannis Kalantidis, Mert Bülent Sariyildiz, Noé Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. In *NeurIPS*, 2020a.

Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. *arXiv preprint arXiv:2010.01028*, 2020b.

Sungnyun Kim, Gihun Lee, Sangmin Bae, and Se-Young Yun. Mixco: Mix-up contrastive learning for visual representation. *arXiv preprint arXiv:2010.06300*, 2020.

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

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

Risi Kondor and Horace Pan. The multiscale laplacian graph kernel. *arXiv preprint arXiv:1603.06186*, 2016.

Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, volume 2002, pp. 315–322, 2002.

Nils Kriege and Petra Mutzel. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483*, 2012.

Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. I-mix: A domain-agnostic strategy for contrastive representation learning. *arXiv preprint arXiv:2010.08887*, 2020.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International Conference on Machine Learning*, pp. 3835–3845. PMLR, 2019.

Aaron F McDaid, Derek Greene, and Neil Hurley. Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint arXiv:1110.2515*, 2011.

Mehryar Mohri and Afshin Rostamizadeh. Rademacher complexity bounds for non-iid processes. 2009.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5115–5124, 2017.

Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pp. 2014–2023. PMLR, 2016.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, 2018.

Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6519–6528, 2019.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 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.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1150–1160, 2020.

Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 287–297. Springer, 2008.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*, 2020.

Jorge M Santos and Mark Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *International conference on artificial neural networks*, pp. 175–184. Springer, 2009.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Zhiqiang Shen, Zechun Liu, Zhuang Liu, Marios Savvides, and Trevor Darrell. Rethinking image mixture for unsupervised visual representation learning. *arXiv e-prints*, pp. arXiv–2003, 2020.

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pp. 488–495. PMLR, 2009.

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Mahito Sugiyama, M Elisabetta Ghisu, Felipe Llinares-López, and Karsten Borgwardt. graphkernels: R and python packages for graph comparison. *Bioinformatics*, 34(3):530–532, 2018.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *Proceedings of the 2018 world wide web conference*, pp. 539–548, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *stat*, 1050:21, 2018.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR (Poster)*, 2019.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pp. 6438–6447. PMLR, 2019.

Vikas Verma, Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc Le. Towards domain-agnostic contrastive learning. In *International Conference on Machine Learning*, pp. 10530–10541. PMLR, 2021.

Sheng Wan, Shirui Pan, Jian Yang, and Chen Gong. Contrastive and generative graph convolutional networks for graph-based semi-supervised learning. *arXiv preprint arXiv:2009.07111*, 2020.

Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 889–898, 2017.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere, 2020.

Xiangyu Wang, Chenlei Leng, and David B Dunson. On the consistency theory of high dimensional variable screening. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/540ae6b0f6ac6e155062f3dd4f0b2b01-Paper.pdf.

Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural networks: Tricks of the trade*, pp. 639–655. Springer, 2012.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018a.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5453–5462. PMLR, 2018b.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33, 2020.

Zhiping Zeng, Anthony KH Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1):25–36, 2009.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Qiang Zhou, Chaohui Yu, Zhibin Wang, Qi Qian, and Hao Li. Instant-teaching: An end-to-end semi-supervised object detection framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4081–4090, 2021.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *WWW*, 2021.

Table 7: Statistics of classification datasets. For graph classification dataset, # NODES, # EDGES imply average number of nodes and edges in each graph.

statistics	node classification					graph classification				
	Citeseer	Cora	Pubmed	DBLP	PPI	MUTAG	PTC_MR	IMDB-BIN	IMDB-MUL	REDDIT-BIN
# graphs	1	1	1	1	24	188	344	1000	1500	2000
# nodes	3327	2708	19717	17716	56944	17.93	14.29	19.77	13.0	508.52
# edges	4732	5429	105734	44338	818716	19.79	14.69	193.06	65.93	497.75
# classes	6	7	3	4	121	2	2	2	3	2

A Implementation Details

Dataset statistic information. For node classification in transductive learning, we use Cora[4], Citeseer[5], Pubmed[6] and DBLP citation networks (Sen et al., 2008) where documents (nodes) are connected through citation relations (edges). For graph classification, we use the following datasets: MUTAG (Kriege & Mutzel, 2012) containing MUTAGenic compounds, PTC_MR (Kriege & Mutzel, 2012) containing 344 chemical compounds represented as graphs which report the carcinogenicity for rats, IMDB-BINARY and IMDB-MULTI (Yanardag & Vishwanathan, 2015) containing 1,000 actors/actresses who played roles in movies in IMDB. In each graph, nodes represent actors/actresses, and there is an edge between them if they appear in the same movie. These graphs are derived from the Action and Romance genres. All the graph datasets can be downloaded on this site [7]. Last, we predict protein roles in inductive learning protocol, in terms of their cellular functions from gene ontology, within the protein-protein interaction (PPI) network to evaluate the generalization ability of the proposed m-mix across multiple graphs. The PPI dataset contains multiple graphs, with each corresponding to human tissue. The graphs are constructed by (Hamilton et al., 2017), where each node has multiple labels that are a subset of gene ontology sets (121 in total), and node features include positional gene sets, motif gene sets, and immunological signatures (50 in total). Following previous methods (Zhu et al., 2020; Hamilton et al., 2017), we select twenty graphs consisting of 44,906 nodes as the training set, two graphs containing 6,514 nodes as the validation, and the rest four graphs containing 12,038 nodes as the test set. The detailed statistic information can be viewed in Table 7.

Hyper-parameter details. We choose GCN (Kipf & Welling, 2016) as our backbones, whose parameters are initialized through Xavier initialization (Glorot & Bengio, 2010). The model is trained using Adam (Kingma & Ba, 2014) with a learning rate of 3e-4 and τ is set 1. Instead of using grid search to select the optimal number of GCN layers in different datasets (Hassani & Khasahmadi, 2020; Sun et al., 2019), our method only uses a single-layer network as the backbone. For graph classification, We search $BatchSize$ from [256, 512, 1024, 2048], which can be trained on a single GPU. For node classification, we follow DGI (Velickovic et al., 2018) and set the number of epochs to 2000 and select $BatchSize$ from [2, 4, 8] (note that on large graphs, we have to sample some subgraphs instead of using the full graph for training). We also use early stopping with the patience of 30 to prevent overfitting. We search the dimension of hidden space of both node and graph representations from [32, 64, $\cdots$, 2048]. For mo-mix, we set C as 0.2 as default.

Evaluation Protocol. In line with (Hassani & Khasahmadi, 2020; Park et al., 2019), we evaluate our approach under the linear evaluation protocol for both node and graph classification. For node classification, we report the mean classification accuracy with standard deviation on the test nodes after 50 runs of training followed by a linear classifier. For graph classification, we follow the standard protocol in InfoGraph (Sun et al., 2019) and report the mean 10-fold cross-validation accuracy with standard deviation after 5 runs followed by a linear SVM (Hearst et al., 1998). We also conduct experiments on two downstream tasks: graph edit distance and node clustering, which we discuss in Appendix B.

Implementation details of baseline MVGRL and GRACE. For MVGRL+(mp-mix), in line with (Hassani & Khasahmadi, 2020), for a given dataset, the node features are processed as follows. If the dataset contains initial node features, they are normalized using the standard score. If the dataset does

[4] https://relational.fit.cvut.cz/dataset/Cora
[5] http://networkrepository.com/Citeseer.php
[6] https://deepai.org/dataset/Pubmed
[7] https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets

Table 8: Normalized MI (NMI) and adjusted rand index (ARI) score on node clustering. The score is directly calculated by scikit-learn.

	method	Cora		Citeseer		Pubmed	
		NMI	ARI	NMI	ARI	NMI	ARI
unsupervised	VGAE (Kipf & Welling, 2016)	0.3292	0.2547	0.2605	0.2056	0.3108	0.3018
	MGAE (Wang et al., 2017)	0.5111	0.4447	0.4122	0.4137	0.2822	0.2483
	ARGA (Pan et al., 2018)	0.4490	0.3520	0.3500	0.3410	0.2757	0.2910
	ARVGA (Pan et al., 2018)	0.4500	0.3740	0.2610	0.2450	0.1169	0.0777
	GALA (Park et al., 2019)	0.5767	0.5315	0.4411	0.4460	0.3273	0.3214
	MVGRL (Hassani & Khasahmadi, 2020)	0.6291	0.5696	0.4696	0.4497	0.3609	0.3386
	MVGRL+ (mp-mix)	**0.6483**	**0.5917**	**0.4736**	**0.4628**	**0.3704**	**0.3431**

not contain initial node features but carries node labels, they are used as the initial node features (only in graph classification datasets). Otherwise, if the dataset has neither, the node features are initialized with node degrees. For GRACE+(mp-mix), we adopt the same augmentation in GRACE (Zhu et al., 2020), with 0.1 and 0.0 probability masking features in view (a) and (b), respectively, and dropping edges with 0.1 and 0.4 probability on DBLP. For the PPI dataset, we drop edges with 0.3, 0.4 probability and mask features with 0.1 and 0.0 probability.

Detail of ablation experiments on different sample sizes and contrastive dimension. We test our method under different sample sizes and contrastive dimension. We set the epoch as 2000 and record the accuracy every 10 epochs. We find with a higher contrastive dimension, the model's stability is increasing, meanwhile, the convergence rate is faster than the low dimension. When set contrastive dimension as 2048, the best model can be obtained in 200 epochs, while set hidden dimension as 64, the convergence rate of the model is much slower (best results are got about 2000 epochs). From Figure 3, we can easily observe that with a large contrastive dimension, initialed embedding has remarkable representation ability. We guess it is because, with a larger contrastive dimension, the initialized node embedding vector is closer to the optimal solution (orthogonal in hypersphere).

B Additional Experiments

Cluster setup and results. Followed by (Hassani & Khasahmadi, 2020), we evaluate our method under clustering evaluation protocol and cluster the learned representations using the K-Means algorithm for node classification. In detail, we set the number of clusters to the number of ground-truth classes and report the average normalized mutual information score (NMI) (McDaid et al., 2011) and adjusted rand score (ARI) (Santos & Embrechts, 2009).

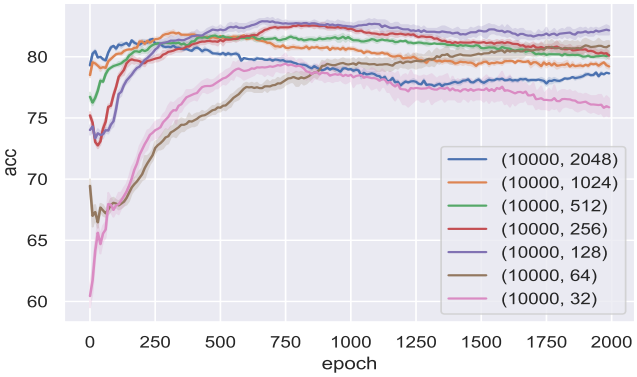


Figure 3: Accuracy with different batch / hidden pairs over training epochs. For instance, $(10000, 2048)$ means training with $N = 10000$, $\mathcal{F}' = 2048$.

Details of graph edit distance. Graph edit distance between graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ is defined as the minimum number of edit operations needed to transform $\mathcal{G}_1$ to $\mathcal{G}_2$. Typically the edit operations include add / remove / substitute nodes and edges. However computing the graph edit distance is NP-complete problem in general (Zeng et al., 2009), therefore approximations have to be used. There's also some attempts by deep graph model (Li et al., 2019). In detail, they construct triplet pairs through edit graph (substitute and remove) edges, which is also an unsupervised model. For instance, they substitute k_p edges from graph $\mathcal{G}_1$ to generate $\mathcal{G}_{1p}$, then substitute k_n edges to generate $\mathcal{G}_{1n}$. By set $k_p < k_n$, then, they regard the graph edit distance between $(\mathcal{G}_1, \mathcal{G}_{1p})$ is shorter than $(\mathcal{G}_1, \mathcal{G}_{1n})$. But actually, the edit-distance

Table 9: Performance on graph classification datasets with different objective functions.

METHODS	MUTAG	PTC_MR	IMDB-BIN	IMDB-MULTI	REDDIT-BIN
mp-mix ($\mathcal{J}_{node}$ only)	91.4 ± 0.4	63.7 ± 0.4	74.9 ± 0.6	51.6 ± 0.5	88.4 ± 0.3
mp-mix ($\mathcal{J}_{graph}$ only)	89.4 ± 0.7	61.0 ± 0.6	73.5 ± 0.6	50.7 ± 0.4	84.3 ± 0.7
mp-mix (w/o $\mathcal{L}_{div}$)	90.7 ± 1.3	63.7 ± 0.9	75.1 ± 0.7	52.0 ± 0.7	88.1 ± 0.6
mp-mix	$\mathbf{92.0 \pm 0.5}$	$\mathbf{64.6 \pm 0.4}$	$\mathbf{75.6 \pm 0.5}$	$\mathbf{52.9 \pm 0.4}$	$\mathbf{88.7 \pm 0.4}$

Table 10: Performance on graph and node classification datasets of different branches. $\mathbf{z}^a$, $\mathbf{h}^a$ means only use the attentive branch output as the final output. Average is the average ensemble, which means $\mathbf{z} = (\mathbf{z}^a + \mathbf{z}^a)/2$ and $\mathbf{h} = (\mathbf{h}^b + \mathbf{h}^b)/2$. Same with the ensemble property, the aggregated output is more stable than individual output.

Branch	Cora	Citeseer	Pubmed	MUTAG	PTC_MR	IMDB-BIN	IMDB-MULTI
$\mathbf{z}^a$, $\mathbf{h}^a$	85.3 ± 0.3	75.6 ± 0.3	82.7 ± 0.4	91.1 ± 0.5	64.5 ± 0.5	75.5 ± 0.7	52.6 ± 0.6
$\mathbf{z}^b$, $\mathbf{h}^b$	85.5 ± 0.4	75.5 ± 0.4	82.7 ± 0.5	91.8 ± 0.6	64.1 ± 0.5	75.1 ± 0.6	52.4 ± 0.5
Average	$\mathbf{85.9 \pm 0.3}$	$\mathbf{75.8 \pm 0.3}$	$\mathbf{82.9 \pm 0.3}$	$\mathbf{92.0 \pm 0.5}$	$\mathbf{64.6 \pm 0.4}$	$\mathbf{75.6 \pm 0.5}$	$\mathbf{52.9 \pm 0.4}$

between $(\mathcal{G}_1, \mathcal{G}_{1p})$ can be smaller than $(\mathcal{G}_1, \mathcal{G}_{1n})$ due to symmetry and isomorphism. However, the probability of such cases is typically low and decreases rapidly with increasing graph sizes. Finally, they train their model on these triplet pairs and evaluate other triplets. Simpler than theirs, we don't need to design special pretext tasks on different tasks, but only use the pre-trained model to evaluate this task.

Effect of graph level contrastive loss. Graph classification usually requires global information of one graph, where node level contrastive learning methods usually learn local information. To further explore the effect of contrastive learning in different levels. We conduct experiments with graph-level only, node-level only. Table 9 shows $\mathcal{J}_{graph}$ can helpfully improve graph level representation, while without $\mathcal{J}_{node}$, GNN encoder can not extract fine-grained information.

Effect of architecture aggregating. To demonstrate our performance is not due to GNN architecture. We reported the accuracy of two branches (view (a) and view (b)), note that the difference between the two branches is only augmentation hyper-parameters. As Table 10 shows, both two branches achieve remarkable results.

More discussion of m-mix. Based on multiple samples mixing on graph learning and vision, our method can be further applied in the language domain. We think the revised module is an interesting direction to explore. Except using prior knowledge and using threshold, we can also mix by clustering-based methods, i.e., using clustering methods to construct image-level adjacency matrix, which may generate more useful hard negatives. We leave such ideas for future work.

C PROOFS

Proof of Proposition 1

Proof *Given two negative pairs $(\mathbf{z}_i, \mathbf{z}_j)$ and $(\mathbf{z}_i, \mathbf{z}_k)$, where $\mathbf{z}_i$ is the i-th sample's embedding. Denote λ_i as the mixing weight of $\mathbf{z}_i$. For simplicity, we consider the dot product as the similarity metric function. For multiple instance mixing, we have that $\hat{\mathbf{z}}_i = \sum_i \lambda_i \mathbf{z}_i$. Then, the similarity of mixed sample and original can be written as:*

$$\mathcal{H}(\hat{\mathbf{z}}_i, \mathbf{z}_j) = (\sum_i \lambda_i \mathbf{z}_i)^\top \mathbf{z}_j = \sum_{i \neq j, i \neq k} (\lambda_i \cdot \mathbf{z}_i^\top \mathbf{z}_j) + \lambda_j \cdot \mathbf{z}_j^\top \mathbf{z}_j + \lambda_k \mathbf{z}_k^\top \mathbf{z}_j \tag{12}$$

where the $\sum_{i \neq j, i \neq k} (\lambda_i \cdot \mathbf{z}_i^\top \mathbf{z}_j)$ is irrelevant to negative pairs $(\mathbf{z}_i, \mathbf{z}_j)$ and $(\mathbf{z}_i, \mathbf{z}_k)$. Thus, we assume $\sum_{\lambda_i, i \neq j, i \neq k}$ is a constant. Now we consider another group of mixing weights $(\lambda'_1, \cdots, \lambda'_N)$, which satisfies $\lambda_j = \lambda'_k$ and $\lambda_k = \lambda'_j$ (Note that the reason why we consider this condition is for any group

(λ'_i), we can find the corresponding group (λ_i), which can satisfy this condition). We have:

$$\begin{aligned}\mathcal{H}(\hat{\mathbf{z}}'_i, \mathbf{z}_j) &= (\sum_i \lambda'_i \mathbf{z}_i)^\top \mathbf{z}_j = \sum_{i\neq j, i\neq k} (\lambda'_i \cdot \mathbf{z}_i^\top \mathbf{z}_j) + \lambda'_j \cdot \mathbf{z}_j^\top \mathbf{z}_j + \lambda'_k \mathbf{z}_k^\top \mathbf{z}_j \\ \mathcal{H}(\hat{\mathbf{z}}'_i, \mathbf{z}_k) &= (\sum_i \lambda'_i \mathbf{z}_i)^\top \mathbf{z}_k = \sum_{i\neq j, i\neq k} (\lambda'_i \cdot \mathbf{z}_i^\top \mathbf{z}_k) + \lambda'_j \cdot \mathbf{z}_j^\top \mathbf{z}_k + \lambda'_k \mathbf{z}_k^\top \mathbf{z}_k\end{aligned} \tag{13}$$

Recall the $\mathbf{z}$ are l_2 normalized embedding, i.e., $\mathbf{z}^\top \mathbf{z} = 1$. Thus, we can obtain $\mathcal{H}(\hat{\mathbf{z}}_i, \mathbf{z}_j) > \mathcal{H}(\hat{\mathbf{z}}'_i, \mathbf{z}_j)$. For $\mathcal{H}(\hat{\mathbf{z}}_i, \mathbf{z}_j)$ and $\mathcal{H}(\hat{\mathbf{z}}'_i, \mathbf{z}_k)$, since we have $\lambda_j = \lambda'_k$ and $\lambda_j > \lambda_k$, we can get $\lambda_k \mathbf{z}_k^\top \mathbf{z}_j > \lambda'_j \cdot \mathbf{z}_j^\top \mathbf{z}_k$. Finally, we can get $\mathcal{H}(\hat{\mathbf{z}}_i, \mathbf{z}_j) > max(\mathcal{H}(\hat{\mathbf{z}}'_i, \mathbf{z}_j), \mathcal{H}(\hat{\mathbf{z}}'_i, \mathbf{z}_k))$, i.e., we generate a more difficult negative pair by assigning similar instances with larger mixing weights.

Proof of Theorem 1.

Proof *Since we discard the projection head, we consider the empirical Rademacher complexity in a linear classification hypothesis $h(\mathbf{z}) = \mathbf{w}^\top \mathbf{z}$. For original data, the Rademacher complexity can be written as:*

$$\begin{aligned}\mathfrak{R}_{\mathcal{Z}}(\mathcal{H}_f) &= \mathbb{E}_\sigma \left[\frac{1}{N} \underset{\|\mathbf{w}\|_2 \leq \Theta}{sup} \sum_{i=1}^N \sigma_i \mathbf{w}^\top \mathbf{z}_i \right] = \mathbb{E}_\sigma \left[\frac{1}{N} \underset{\|\mathbf{w}\|_2 \leq \Theta}{sup} \mathbf{w}^\top \sum_{i=1}^N \sigma_i \mathbf{z}_i \right] \\ &= \frac{1}{N} \mathbb{E}_\sigma \left[\underset{\|\mathbf{w}\|_2 \leq \Theta}{sup} \mathbf{w}^\top \sum_{i=1}^N \sigma_i \mathbf{z}_i \right] = \frac{1}{N} \mathbb{E}_\sigma \left[\Theta \| \sum_{i=1}^N \sigma_i \mathbf{z}_i \|_2 \right] \\ &\leq \frac{\Theta}{N} \left(\mathbb{E}_\sigma \left[\| \sum_{i=1}^N \sigma_i \mathbf{z}_i \|_2^2 \right] \right)^{\frac{1}{2}} = \frac{\Theta}{N} (\sum_{i=1}^N \|\mathbf{z}_i\|_2)^{\frac{1}{2}}\end{aligned} \tag{14}$$

where Θ is a constant that regularizes the l_2 norm of the weight vector. Then, we introduce Rademacher complexity $\mathfrak{R}^b_{\mathcal{Z}}(\mathcal{H}_f)$ of binary mixed data as intervening variable, where the generated data can be written as: $\mathbf{z}'_i = \lambda \mathbf{z}_i + (1-\lambda)\mathbf{z}_j$. Then, we have:

$$\begin{aligned}\mathfrak{R}^b_{\mathcal{Z}}(\mathcal{H}_f) &\leq \frac{\Theta}{N} \left(\sum_{i=1}^N \|\mathbf{z}'_i\|_2^2 \right)^{\frac{1}{2}} = \frac{\Theta}{N} \left(\sum_{i=1}^N \left\| \mathbb{E}_{\mathbf{z}_j} \left[\lambda \mathbf{z}_i + (1-\lambda)\mathbf{z}_j \right] \right\|_2^2 \right) \\ &= \frac{\Theta}{N} \left(\sum_{i=1}^N \left\| \lambda \mathbf{z}_i + (1-\lambda) \mathbb{E}_{\mathbf{z}_j} \left[\mathbf{z}_j \right] \right\|_2^2 \right)^{\frac{1}{2}} \leq \frac{\Theta}{N} \left(\sum_{i=1}^N \left(\left\| \lambda \mathbf{z}_i \right\|_2^2 + \left\| (1-\lambda) \mathbb{E}_{\mathbf{z}_j} [\mathbf{z}_j] \right\|_2^2 \right) \right)^{\frac{1}{2}} \\ &= \frac{\Theta}{N} \left(\lambda^2 \sum_{i=1}^N \|\mathbf{z}_i\|_2^2 + (1-\lambda)^2 \sum_{i=1}^N \left\| \mathbb{E}_{\mathbf{z}_j} [\mathbf{z}_j] \right\|_2^2 \right)^{\frac{1}{2}}\end{aligned} \tag{15}$$

where $\mathbb{E}_{\mathbf{z}_j}[\lambda \mathbf{z}_i + (1-\lambda)\mathbf{z}_j]$ is the expectation of linear combination of input data by binary mixup. Then, we give the Rademacher complexity of m-mixed data:

$$\mathfrak{R}^m_{\mathcal{Z}}(\mathcal{H}_f) \leq \frac{\Theta}{N} (\sum_{i=1}^N \|\mathbf{z}'_i\|_2^2)^{\frac{1}{2}} = \frac{\Theta}{N} \left(\sum_{i=1}^N \mathbb{E}_\lambda \left\| \lambda_i \mathbf{z}_i + \sum_j \lambda_j \mathbf{z}_j \right\|_2^2 \right) \tag{16}$$

where λ_i is pre-defined in $mo{-}mix$, and we can set $\lambda = \lambda_i$ to simplify the proof. Compare Eq. 16 with Eq. 15, we can find the different term is mixing term in RHS. Fix the first term, since all the embeddings $\mathbf{z}$ are in l_2 normalization, then, it holds $\mathbb{E}_\lambda \| \sum_j \lambda_j \mathbf{z}_j \|_2^2 \leq \left\| (1-\lambda) \mathbb{E}_{\mathbf{z}_j} [\mathbf{z}_j] \right\|_2^2$. Then, combining Eq. 14, Eq. 15 and Eq.16, we can get:

$$\begin{aligned}\mathfrak{R}_{\mathcal{Z}}(\mathcal{H}_f) - \mathfrak{R}^m_{\mathcal{Z}}(\mathcal{H}_f) &\leq \frac{\Theta \sqrt{1-\lambda_i^2}}{N} (\sum_{i=1}^N \|\mathbf{z}_i\|_2^2 - \sum_{i=1}^N \|\mathbb{E}_{\mathbf{z}_j} [\mathbf{z}_j]_2^2)^{\frac{1}{2}} \\ &= \frac{\Theta \sqrt{1-\lambda_i^2}}{\sqrt{N}} \left(s^2 \left(\|\mathbf{z}_i\|_2 \right) + \|\bar{\mathbf{z}}\|_2^2 - \|\bar{\mathbf{z}}\|_2^2 \right)^{\frac{1}{2}} = \frac{\Theta \sqrt{1-\lambda_i^2}}{\sqrt{N}} \sqrt{s^2 \|\mathbf{z}\|_2} \geq 0\end{aligned} \tag{17}$$

where s is the variance of original data. Note that above equation holds when $\lambda_i = min\{\lambda_1, \lambda_2, \cdots, \lambda_N | \lambda_i \neq 0\}$. Then, we can complete the proof. □

Proof of Theorem 2.

Proof *Split the infoNCE by numerator and denominator parts, we have:*

$$\mathcal{L}_{info} = \mathbb{E}_{(x,y)\sim p_{pos}}[-f(x)^\top f(y)/\tau] + \mathbb{E}_{\substack{(x,y)\sim p_{pos} \\ \{x_i^-\}_{i=1}^N \sim p_{data}}} \left[\log(e^{f(x)^\top f(y)/\tau} + \sum_i e^{f(x)^\top f(x_i^-)/\tau})\right] \tag{18}$$

Followed by (Wang & Isola, 2020), the $\mathcal{L}_{info}$ is equal to pulling embedding to $\mathcal{F}'$ hyper-sphere uniformly. We consider contrastive embedding in both orthogonal solution (optimal) and non-orthogonal solution. For orthogonal solution, we have $\mathbb{I}_{i\neq j}\mathbf{Z}_i^\top \mathbf{Z}_j = 0$ and $\{\mathbf{Z}_i^\top \mathbf{Z}_i = 1\}_{i=0}^N$. For linear classification or linear regression, we have $\mathcal{W} = \mathbf{Z}^\top(\mathbf{Z}\mathbf{Z}^\top)^{-1}$. Take $\mathcal{W}$ to Φ, we can obtain $\Phi = I_{\mathcal{F}'} - 2\tau^{-1}\eta I_{\mathcal{F}'}$, where Φ is the diagonal matrix, then, we can complete the proof. Consider in non-orthogonal solution, if we have:

$$\min_i |\Phi_{ii}| > 2s\rho \max_{ij} |\Phi| + 2\tau^{-1}\|\mathbf{Z}^\top(\mathbf{Z}\mathbf{Z}^\top)^{-1}\epsilon\|_\infty \tag{19}$$

Then, the proof is finished because $\Phi - 2\tau^{-1}\|\mathbf{Z}^\top(\mathbf{Z}\mathbf{Z}^\top)^{-1}\epsilon\|_\infty$ is already a restricted diagonally dominant matrix. From Lemma 3 and 4 of (Wang et al., 2015), we can obtain a union bounds:

$$P(\max_{i\neq j} |\Phi| > c_4 kt\frac{\sqrt{N}}{\mathcal{F}'}) \geq 5(p^2-p)^{-CN} + 2(p^2-p)^{-t^2/2} \tag{20}$$

and

$$P(\|\mathcal{W}\epsilon\|)_\infty \leq \frac{2\sigma\sqrt{c_2}kt\sqrt{N}}{(1-c_0^{-1})\mathcal{F}'} < 4\mathcal{F}'e^{-CN} + 2\mathcal{F}'e^{-t^2/2} \tag{21}$$

where t is any constant which satisfies $t > 0$, c_i is some constants which satisfy $c_0 > 1$, $0 < c_1 < 1 < c_2$ and $c_3 > 0$. k is the conditional number of $\mathbf{Z}$. Then, let $t = \sqrt{CN}/\mu$ for $\mu > 0$, which satisfies the definition of $t > 0$. We can obtain:

$$\frac{N}{\mathcal{F}'}(c_1 k^{-1} - \frac{2c_4\sqrt{C}ks\rho}{\mu} - \frac{2\sigma\sqrt{c_2 C}k}{1-c_0^{-1}\tau\mu}) > 0 \tag{22}$$

where τ/σ evaluates the signal-to-noise ratio. Take μ as the variable, we can derive $\mu > \frac{2c_4\sqrt{C}k^2\rho s}{c_1} + \frac{2\sigma\sqrt{c_2 C}k^2}{c_1\tau(1-c_0^{-1})}$, where the RHS is larger than 1, i.e., $\mu > 1$. Then we have:

$$P_{non-orth} < (p+5p^2)e^{-CN} + 2p^2 e^{-CN/\mu} \tag{23}$$

Recall we assume $\mathcal{F}' > N$ and prove $\mu > 1$, then we can complete the proof for any $\delta > 0$, there's at least $1-\delta$ satisfies $N \geq \frac{\mu^2}{C}(\log(7+1/N) + 2\log\mathcal{F}' - \log\delta)$. □

D VISUALIZATION

Our m-mix can adaptively select which samples should be mixed and gives them different mixing weights. We randomly select anchor sample in CIFAR-10 and CIFAR-100 datasets, and visualize the mixing samples selected by m-mix in Fig. 4 and Fig. 5

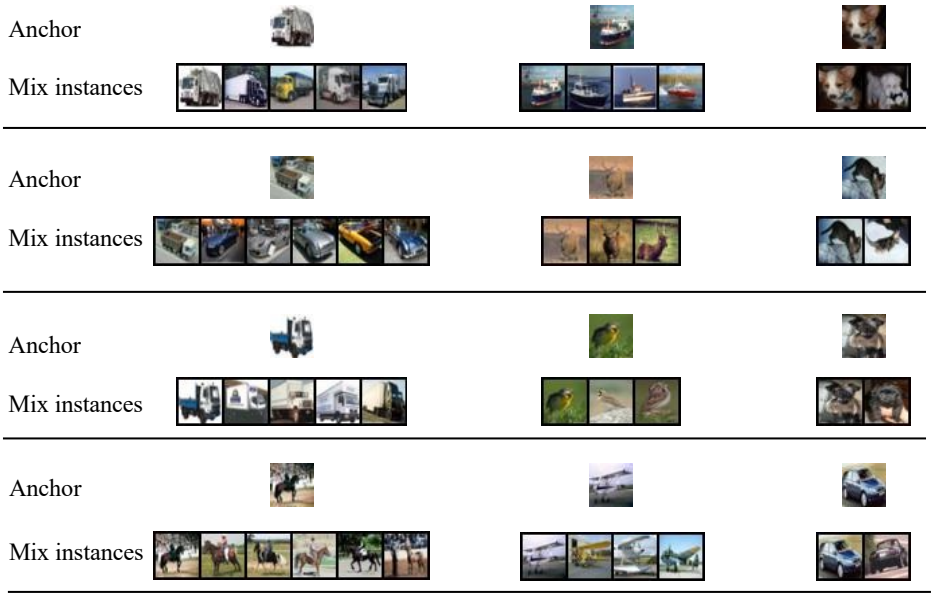


Figure 4: Visualization of multiple samples mixing on cifar-10, where anchor means the input query image and mix samples mean the selected mixing instances by m-mix.

Figure 5: Visualization of multiple samples mixing on cifar-100.