M-Mix: Generating Hard Negatives via Multi-sample Mixing for Contrastive Learning

Shaofeng Zhang
Shanghai Jiao Tong University
Shanghai, China
sherrylone@sjtu.edu.cn

Meng Liu
Shanghai Jiao Tong University
Shanghai, China
meng.liu@sjtu.edu.cn

Junchi Yan[*]
Shanghai Jiao Tong University
Shanghai, China
yanjunchi@sjtu.edu.cn

Hengrui Zhang
Shanghai Jiao Tong University
Shanghai, China
sqstardust@gmail.com

Lingxiao Huang
Huawei TCS Lab
Shanghai, China
huanglingxiao2@huawei.com

Xiaokang Yang
Shanghai Jiao Tong University
Shanghai, China
xkyang@sjtu.edu.cn

Pinyan Lu
Shanghai University of Finance and Economics
Huawei TCS Lab
Shanghai, China
Lu.pinyan@mail.shufe.edu.cn

ABSTRACT

Negative pairs, especially hard negatives as combined with common negatives (easy to discriminate), are essential in contrastive learning, which plays a role of avoiding degenerate solutions in the sense of constant representation across different instances. Inspired by recent hard negative mining methods via pairwise mixup operation in vision, we propose M-Mix, which dynamically generates a sequence of hard negatives. Compared with previous methods, M-Mix mainly has three features: 1) adaptively choose samples to mix; 2) simultaneously mix multiple samples; 3) automatically assign different mixing weights to the selected samples. We evaluate our method on two image datasets (CIFAR-10, CIFAR-100), five node classification datasets (PPI, DBLP, Pubmed, etc), five graph classification datasets (IMDB, PTC_MR, etc), and two downstream combinatorial tasks (graph edit distance and node clustering). Results show that it achieves state-of-the-art performance under self-supervised settings. Code is available at: https://github.com/Sherrylone/m-mix.

CCS CONCEPTS

• **Computing methodologies → Mixture modeling**.

KEYWORDS

Self-supervised learning, Graph neural network, Contrastive learning, Hard sample mining

ACM Reference Format:
Shaofeng Zhang, Meng Liu, Junchi Yan, Hengrui Zhang, Lingxiao Huang, Xiaokang Yang, and Pinyan Lu. 2022. M-Mix: Generating Hard Negatives via Multi-sample Mixing for Contrastive Learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3534678.3539248

1 INTRODUCTION

Graph neural networks (GNNs) [21, 45] 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, the classic GNN training [21] calculates the objective function only from the costly and limited labeled nodes, ignoring the information contained in large amounts of unlabeled nodes. To address this issue, recent studies [12, 53, 61] introduce contrastive learning [13, 57, 58] into self-supervised learning on graphs. In general, graph-level contrastive methods [4] require a large number of negative samples to avoid degenerate solutions (i.e. the trained model projects all samples the identical representation [13]) and boost the performance. However, such a large number of negatives are computational expensive and hard to store. Fortunately, mining or generating a compact set of hard negatives [19, 23] is an effective way to reduce the number of negatives and improve the prediction accuracy, which is important for contrastive learning.

Existing hard negative mining methods are almost from computer vision, and they can be generally divided into two categories: **(1)** Adversarial based methods [17] and **(2)** Mixing based methods [18, 19, 23, 44]. Adversarial-based methods update negative

[*] S. Zhang, M. Liu, J. Yan, X. Yang are also with MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, China. Junchi Yan is the correspondence author. This work was supported by National Key Research and Development Program of China (2020AAA0107600), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), NSFC (U19B2035), and fund from Huawei Inc.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539248

samples before updating encoder networks [14], i.e., maximize the similarity between negative pairs before updating encoders, and such a strategy is basically inspired by adversarial training [8]. Mixing-based methods are inspired by classical data augmentation methods namely mixup [55], which in general is used to help classify samples close to the boundary. In contrastive learning, mixing based methods generate hard negatives by mixing the positive samples and negative samples with pre-defined mixing weights, which can mainly lead to two problems: **1)** due to sampling the mixed negative samples randomly [19] and using static pre-defined mixing weight, the information of similarity between two samples will be ignored[1] **2)** Mix operation is conducted between every two samples, limiting the generated negatives' difficulty for contrastive learning. In this paper, we propose M-Mix, which generates hard negatives via mixing more than two samples with different weights. The hope is to expand the mixing scheme for higher difficulty.

The highlights of this paper are summarized as follows.

1) We propose M-Mix to mine hard negatives, which mixes multiple samples and assigns different mixing weights dynamically. To our best knowledge, this is the first attempt to mix multiple samples (beyond two samples) in contrastive learning.

2) We propose to emphasize mixing weights between similar samples whose efficacy of generating more difficult hard negatives than by random, is grounded by our theoretical analysis (see Theorem 3.1). Specifically, we devise a diversity objective function, to increase the difficulty of generated negatives, which is empirically shown can meanwhile improve the stability in terms of prediction.

3) We further design two modules to complete the mixing method: named M-Mix-wp and M-Mix-op, where the former utilizes the structural information (adjacency matrix) for denoising and increasing prediction accuracy for graph-related tasks. The latter does not require structural information, which can be applied in vision to improve the representation ability.

4) We empirically show that it causes little accuracy drop on graph data, by discarding the non-linear projection head [4], and performing direct contrasts in output space. We further theoretically show (see Theorem 3.2) the efficacy of this simplification and particularly both the analysis and empirical sensitivity study of different combinations of sample size and output dimension.

5) We adapt and incorporate M-Mix into the frameworks of both SimCLR [4] and MVGRL [12], to enhance their negative sampling for training, and extensive experimental results on both vision and graph datasets show that the resulting approaches outperform most of the state-of-the-art methods in self-supervised learning. The source code and trained models will be made public available.

2 RELATED WORK

This paper explores hard negatives mining by mixing multiple samples, especially in the context of contrastive learning. We discuss recent progress in contrastive learning in both vision and graph domains, as well as hard negatives mining methods.

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 [27] proposes InfoNCE objective to discriminate positive pairs and negative pairs from sequential data. CMC [38] generalizes CPC to multi-view settings, and DIM [16] 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 [4, 16]. To address the issues of storing a large number of negative samples, MoCo [13] adopts the memory bank strategy and uses a momentum-based technique to update two encoders asynchronously. SimCLR [4] 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 [9], or hard negatives [17].

Hard negative mining in contrastive learning. Hard negatives mining refers to generating negative pairs, which are difficult to discriminate. We divide previous hard negatives mining methods into two categories. **1)** Gradient-based. Inspired by adversarial training [8], Adco[17] tries generating hard negatives via adversarial optimization. In detail, the negatives in the memory bank [13] are first optimized by maximizing contrastive objective, then the encoder network [14] is optimized by minimizing contrastive objective, where the two steps run alternately. **2)** Mixing based. Inspired by classical augmentation method Mixup [55] and its variants [33], MoCHi [18] 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 [23] proposes mixing two samples in input space, where they are first mixed before feeding to the encoder. The concurrent work DACL [44] uses the same idea. Further, DACL conducts experiments on both graph and image datasets.

Contrastive learning in graph. Early works including DGI [42] and InfoGraph [37] adopt the idea of local-global contrastive objective [16] to node/graph representation learning respectively by contrasting node-graph pairs. Then, MVGRL [12] uses fixed diffusion operations such as heat kernel and Personalized PageRank to generate views of the original graph. Then, the 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 [31] generates node views through sub-graph sampling with random walks, where the different subgraphs are taken as negatives. Then, GRACE [61] and its variant GCA [62] introduce SimCLR to graph, and different nodes are taken as negatives. Depart the success of hard negative mining in vision, hard negatives in graph contrastive learning has been little explored.

3 THE PROPOSED M-MIX

In this section, we present M-Mix in detail, starting with preliminaries, followed by the two designed modules (M-Mix-op for vision and M-Mix-wp for graph) for generating hard negative samples, as well as the training framework and overall objective function.

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

[1] In Theorem 3.1, we will theoretically show that two samples with larger similarity should be sampled with higher probability and mixed with larger weights [23].

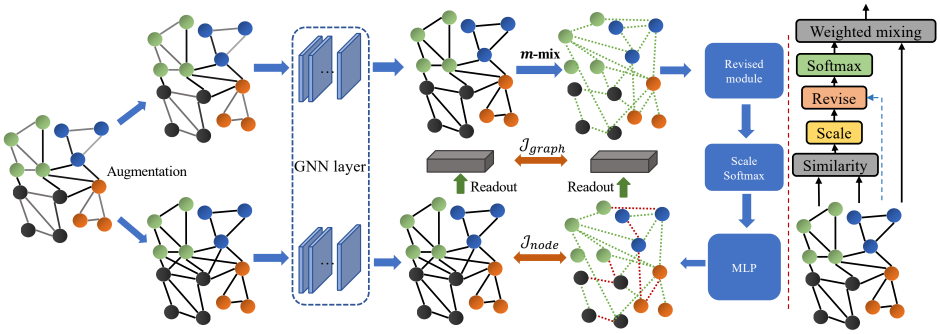

Figure 1: Framework (left) of the proposed M-Mix, where the revised module is discussed in "bridge mixing with prior knowledge" and we provide two different revised modules. Green and red dash lines in left figure denote mixing weights $\lambda_i \neq 0$ and $\lambda_i = 0$, respectively. The blue dash 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. 11.

$\mathbf{A} \in \mathbb{R}^{N\times N}$ 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 [59] and Binary-Mixup [55], 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 [18], we mainly discuss the later kind of mixing.

3.2 Hard Negatives via Multi-Sample Mixing

Multi-sample mixing. Given a set of input 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 multi-sample mixing, we define a set of mixing weights $\lambda = \{\lambda_i\}_{i=1}^N$, and the generated new sample $\hat{\mathbf{z}}_i$ becomes:

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

In the previous methods [18, 55], 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).

Theorem 3.1. ***(Relation of hard negative mining and mixing weights)****. 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 discriminative negative pair.*

Proof of Theorem 3.1

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 multi-sample mixing, we have that $\hat{\mathbf{z}}_i = \sum_i \lambda_i\mathbf{z}_i$. Then, the similarity of mixed sample and the original sample is:

$$\mathcal{H}(\hat{\mathbf{z}}_i, \mathbf{z}_j) = (\sum_i \lambda_i\mathbf{z}_i)^\top\mathbf{z}_j = \sum_{i\neq j, i\neq k} (\lambda_i \cdot \mathbf{z}_i^\top\mathbf{z}_j) + \lambda_j \cdot \mathbf{z}_j^\top\mathbf{z}_j + \lambda_k\mathbf{z}_k^\top\mathbf{z}_j \tag{3}$$

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:

$$\begin{aligned} \mathcal{H}(\hat{\mathbf{z}}'_i, \mathbf{z}_j) &= (\sum_i \lambda'_i\mathbf{z}_i)^\top\mathbf{z}_j = \sum_{i\neq j, i\neq k} (\lambda'_i \cdot \mathbf{z}_i^\top\mathbf{z}_j) + \lambda'_j \cdot \mathbf{z}_j^\top\mathbf{z}_j + \lambda'_k\mathbf{z}_k^\top\mathbf{z}_j \\ \mathcal{H}(\hat{\mathbf{z}}'_i, \mathbf{z}_k) &= (\sum_i \lambda'_i\mathbf{z}_i)^\top\mathbf{z}_k = \sum_{i\neq j, i\neq k} (\lambda'_i \cdot \mathbf{z}_i^\top\mathbf{z}_k) + \lambda'_j \cdot \mathbf{z}_j^\top\mathbf{z}_k + \lambda'_k\mathbf{z}_k^\top\mathbf{z}_k \end{aligned} \tag{4}$$

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. □

Quantification of mixing weights. Theorem 3.1 says similar negatives pairs should be assigned larger mixing weight. 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:

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

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. 5 ignores homogenization [40], 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 be different in many cases [21, 41]. Hence, we propose another approach to quantify mixing weights in two different spaces as follow:

$$\lambda_j = \frac{\exp(\mathcal{H}(\mathbf{z}_i^\top \mathbf{p}_m, \mathbf{z}_j^\top \mathbf{p}_n))}{\sum_k \exp(\mathcal{H}(\mathbf{z}_i^\top \mathbf{p}_m, \mathbf{z}_k^\top \mathbf{p}_n))} \quad s.t. \quad 0 <= j <= N \tag{6}$$

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. 6 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, the mixing weights of other instances are computed as:

$$\lambda_j = \frac{(1-C)\exp(\mathcal{H}(\mathbf{z}_i, \mathbf{z}_j))}{\sum_k \exp(\mathcal{H}(\mathbf{z}_i, \mathbf{z}_k))} \quad s.t. \quad 0 <= j <= N, \ j \neq i \tag{7}$$

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

$$\mathcal{L}_{Div} = -\frac{1}{N}\sum_i \left(\frac{\mathbf{z}_i^\top \mathbf{p}_m}{\|\mathbf{z}_i^\top \mathbf{p}_m\|_2} - \frac{\mathbf{z}_i^\top \mathbf{p}_n}{\|\mathbf{z}_i^\top \mathbf{p}_n\|_2} \right)^2 = 2 \cdot \frac{(\mathbf{z}_i^\top \mathbf{p}_m)^\top (\mathbf{z}_i^\top \mathbf{p}_n)}{\|\mathbf{z}_i^\top \mathbf{p}_m\|_2 \|\mathbf{z}_i^\top \mathbf{p}_n\|_2} - 2 \tag{8}$$

By the diversity objective function, 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 their' 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 [41], 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 objective 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 8.

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 objective is:

$$\begin{aligned} &\mathcal{L}_{info-N}(g(\hat{\mathbf{z}}_i^a)) \\ &= -\log \frac{e^{\mathcal{H}(g(\hat{\mathbf{z}}_i^a), \mathbf{z}_i^b)/\tau}}{\sum_{j=1, j\neq i}^N e^{\mathcal{H}(g(\hat{\mathbf{z}}_i^a), g(\hat{\mathbf{z}}_j^a))/\tau} + \sum_{j=1}^N e^{\mathcal{H}(g(\hat{\mathbf{z}}_i^a), \mathbf{z}_j^b)/\tau}}, \end{aligned} \tag{9}$$

where g denotes the MLP module, as commonly used in previous methods e.g. [9]. The objective is similar to view b. $\mathcal{H}(\cdot, \cdot)$ means the similarity metric function between two vectors (should be the same with $\mathcal{H}$ in Eq. 5 and Eq. 6), and τ denotes the temperature hyper-parameter [4]. The overall objective is formulated as:

$$\mathcal{J}_{node}(\mathcal{G}) = \frac{1}{2N}\sum_i \left[\mathcal{L}_{info-N}(\hat{\mathbf{z}}_i^a) + \mathcal{L}_{info-N}(\mathbf{z}_i^b) \right] + \mathcal{L}_{div} \tag{10}$$

The above objective can capture local information (node-level). While for graph level tasks (graph classification, graph edit distance), we need global information (graph-level). Hence, we design graph-level contrastive objectives. Given M graphs, 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 objective can be formulated as:

$$\mathcal{J}_{graph} = -\frac{1}{M}\left[\sum_{i=1}^M \log \frac{e^{\mathcal{H}(\mathbf{h}_i^a, \mathbf{h}_i^b)}}{\sum_{j, j\neq i} e^{\mathcal{H}(\mathbf{h}_i^a, \mathbf{h}_j^a)} + \sum_j e^{sim(\mathbf{h}_i^a, \mathbf{h}_j^b)}} + \mathcal{J}_{node}(\mathcal{G}_i) \right] \tag{11}$$

3.4 Theoretical Analysis on Dimensions

Since we discard the projection head in our framework, here we give theoretical analysis on how to choose the output dimensions and sample sizes from the screening variables perspective [46].

Theorem 3.2. ***(Suitable output 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 output dimension is large enough in the sense $\mathcal{F}' > N$. When $\mathcal{U} << N < \mathcal{F}'$ which mostly holds in practice, then for any $\delta > 0$ and $C > 0$, if the sample size satisfies:

$$N > \frac{\mu^2}{C}\left(\log\left(7 + \frac{1}{N}\right) + 2\log\mathcal{F}' - \log\delta\right), \tag{12}$$

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 $Z^\top(ZZ^\top)$. ϵ is learned noise in output space.

Proof. Split the infoNCE by numerator and denominator parts:

$$\mathcal{L}_{info} = \mathbb{E}_{(x,y)\sim p_{pos}}[-f(x)^\top f(y)/\tau] + \\ \mathbb{E}_{\substack{(x,y)\sim p_{pos}\\ \{x_i^-\}_{i=1}^N \sim p_{data}}}\left[\log(e^{f(x)\top f(y)/\tau} + \sum_i e^{f(x)^\top f(x_i^-)/\tau})\right] \tag{13}$$

Followed by SimCLR [4], 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:

$$\min_i |\Phi_{ii}| > 2s\rho \max_{ij}|\Phi| + 2\tau^{-1}\|\mathbf{Z}^\top(\mathbf{Z}\mathbf{Z}^\top)^{-1}\epsilon\|_\infty \tag{14}$$

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 [46], we can obtain a union bound:

$$P\left(\max_{i\neq j}|\Phi| > c_4 kt\frac{\sqrt{N}}{\mathcal{F}'}\right) \geq 5(p^2-p)^{-CN} + 2(p^2-p)^{-t^2/2} \tag{15}$$

$$P(\|\mathcal{W}\epsilon\|)_\infty \leq \frac{2\sigma\sqrt{c_2}kt\sqrt{N}}{(1-c_0^{-1})\mathcal{F}'} < 4\mathcal{F}'e^{-CN} + 2\mathcal{F}'e^{-t^2/2} \tag{16}$$

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:

$$\frac{N}{\mathcal{F}'}\left(c_1 k^{-1} - \frac{2c_4\sqrt{C}ks\rho}{\mu} - \frac{2\sigma\sqrt{c_2 C}k}{1-c_0^{-1}\tau\mu}\right) > 0 \tag{17}$$

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:

$$P_{non-orth} < (p+5p^2)e^{-CN} + 2p^2 e^{-CN/\mu} \tag{18}$$

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 $N \geq \frac{\mu^2}{C}(\log(7 + 1/N) + 2\log\mathcal{F}' - \log\delta)$. □

Theorem 3.2 implies that Φ is actually a screening consistent variable (introduced by [46] which helps to select useful feature in large dimension space) for any $\beta \in \mathcal{B}_\tau(s,\rho)$, where $\mathcal{B}_\tau(s,\rho) = \{\beta \in \mathcal{R}^{\mathcal{F}'} : \underset{i\in supp(\beta)}{min}|\beta_i| \geq \tau, supp(\beta) \leq s, \frac{\max_{i\in supp(\beta)}|\beta_i|}{\min_{i\in supp(\beta)}|\beta_i|} \leq \rho\}$ and $supp(f)$ means the support set of f. Note that the set of β is the obtained as a least square solution, i.e., $\beta = \mathcal{W}\mathcal{Y}$, where $\mathcal{Y}$ is the label. We analyze on both optimal solution and non-optimal solution for contrastive learning. We can draw a conclusion for the least square solver, the sample size and output dimension should be at a certain value (not the larger the better). This theoretical result is in fact consistent with our experimental results as will be shown in 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, PTC_MR, IMDB-BINARY, IMDB-MULTI [50] and REDDIT-BINARY [50]. The statistic information are given in Talbe. 1. 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 [61] and diffusion in MVGRL [12]. The model is implemented with PyTorch and each trial is executed on a single Tesla V100 GPU. For image datasets, we mainly evaluate our method on basis of SimCLR on CIFAR-10 and CIFAR-100, where the data augmentation in images is in line with SimCLR [4], i.e., Random Resized Crop to 32 × 32, Random Horizontal Flip, Color Jitter and Gray Scale.

Dataset statistic information. For node classification in transductive learning, we use Cora[2], Citeseer[3], Pubmed[4] and DBLP citation networks where documents (nodes) are connected through citation relations (edges). For graph classification, we use the following datasets: MUTAG containing MUTAGenic compounds, PTC_MR containing 344 chemical compounds represented as graphs which report the carcinogenicity for rats, IMDB-BINARY and IMDB-MULTI 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[5]. 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 [11], 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 [11, 61], 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 is given in Table 1.

Hyper-parameter details. We choose GCN [21] as our backbones. The model is trained using Adam [20] with a learning rate of 3e-4 and τ is set 1. Instead of using grid search to select the optimal

[2] https://relational.fit.cvut.cz/dataset/Cora

[3] http://networkrepository.com/Citeseer.php

[4] https://deepai.org/dataset/Pubmed

[5] https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets

Table 1: Dataset statistics. For graph classification, # 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

number of GCN layers in different datasets [12, 37], 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 [42] and set the number of epochs to 2000 and select *BatchSize* from [2, 4, 8] (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 [12, 29], we evaluate our approach under the linear classifier evaluation protocol for both node and graph level 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 [37] and report the mean 10-fold cross-validation accuracy with standard deviation after 5 runs followed by a linear SVM. We also conduct experiments on two downstream tasks which are known NP-hard: graph edit distance (a measure of similarity between two graphs) and node clustering (nodes with similar attributes have closer distance in embedding space), based on our node embedding.

4.2 Results for Node and Graph Classification

Baselines. We compare with both supervised and unsupervised learning methods. For node classification, we compare our method with recent popular self-supervised methods, e.g. GRAPH-CL [53], MVGRL [12]. The results show our methods (both *mo*-mix and *mp*-mix) achieve state-of-the-art performance on Citeseer and Pubmed. On graph classification tasks, we compare with GCC [31], MVGRL [12] and GRAPH-CL [53]. Experimental results also show that our method outperforms all of the unsupervised methods across datasets except on REDDIT-BINARY. For the results of other baseline models, we quote their results reported by [12].

Results on node classification. The results of node classification in Table 2 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 1.3% 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[6][7]. We also report *mp*-mix without diversity loss, and the results show diversity loss can make M-Mix more stable, also boosts accuracy.

Table 2: Mean accuracy (%) on node classification with 10-fold cross validation. X: node features; A: adjacency matrix; Y: labels; D: diffusion matrix in [38]; S: affinity matrix in this paper; †: our reproduction otherwise results are quoted from original papers. Diff: use diffusion as data augmentation.

	dataset method	input mode	Cora	Citeseer	Pubmed
supervised	MLP [42]	X, Y	55.1	46.5	71.4
	ICA [7]	A,Y	75.1	69.1	73.9
	LP [60]	A,Y	68.0	45.3	63.0
	MANIREG [2]	X, A, Y	59.5	60.1	70.7
	SEMIEMB [2]	X, Y	59.0	59.6	71.7
	PLANETOID [51]	X, Y	75.7	64.7	77.2
	CHEBISHEV [5]	X, A, Y	81.2	69.8	74.4
	GCN [21]	X, A, Y	81.5	70.3	79.0
	MONET [25]	X, A, Y	81.7 ± 0.5	–	78.8 ± 0.3
	JKNET [49]	X, A, Y	82.7 ± 0.4	**73.0 ± 0.5**	77.9 ± 0.4
	GAT [42]	X, A, Y	**83.0 ± 0.7**	72.5 ± 0.7	**79.0 ± 0.3**
unsupervised	LINEAR [42]	X	47.9 ± 0.4	49.3 ± 0.2	69.1 ± 0.3
	DEEPWALK [30]	X,A	70.7 ± 0.6	51.4 ± 0.5	74.3 ± 0.9
	GAE [21]	X,A	71.5 ± 0.4	65.8 ± 0.4	72.1 ± 0.5
	VERSE [39]	X, D, A	72.5 ± 0.3	55.5 ± 0.4	–
	DGI [43]	X, A	82.3 ± 0.6	71.8 ± 0.7	76.8 ± 0.6
	DGI† [43]	X, D	83.5 ± 0.7	71.8 ± 0.4	78.0 ± 0.4
	GraphCL [53]	X, A	82.3 ± 0.1	73.1 ± 0.2	–
	GraphCL† [53]	X, A	83.4 ± 0.7	72.4 ± 0.4	79.4 ± 0.7
	GRACE† [61]	X, A	82.2 ± 0.6	71.6 ± 0.6	78.4 ± 0.5
	InfoGCL [47]	X, A	83.5 ± 0.3	73.5 ± 0.4	79.1 ± 0.2
	MVGRL [12]	X, D, A	**86.8 ± 0.5**	73.3 ± 0.5	80.1 ± 0.7
	MVGRL† [12]	X, D, A	84.1 ± 0.9	73.5 ± 0.6	80.4 ± 1.1
	i-mix [23]	X, A	84.6 ± 0.5	74.1 ± 0.5	81.1 ± 0.2
	CCA-SSG [56]	X, A	84.2 ± 0.4	73.1 ± 0.3	81.6 ± 0.4
ours	*mo*-mix	X, D, A	85.4 ± 0.4	75.1 ± 0.4	82.6 ± 0.2
	mp-mix w/o div	X, D, A	85.1 ± 0.8	74.4 ± 0.9	81.8 ± 0.7
	mp-mix	X, D, A	85.9 ± 0.3	**75.8 ± 0.3**	**82.9 ± 0.3**

Results on graph classification. Table 3 gives classification accuracy on graph, where MVGRL+(*mp*-mix) outperforms the best of existing state-of-the-art methods [12, 31] by 0.8%, 1.1%, 1.5% on MUTAG, PTC_MR, IMDB-MULTI, respectively. On IMDB-BIN and REDDIT-BIN, *mp*-mix achieves 1.2% and 3.6% accuracy than baseline MVGRL.

4.3 Transferability Test on Downstream Tasks

Graph edit distance (GED). i) setup. 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 [54], therefore approximations have to be used. There are also some attempts by deep graph model [24]. In detail, they construct triplet

[6] https://github.com/kavehhassani/mvgrl/issues/2

[7] https://github.com/hengruizhang98/mvgrl

Table 3: Accuracy (%) on graph classification with 10-fold cross validation. RANDOM: Random walk; N2VEC, S2VEC, G2VEC: Node/sub-graph/graph to vector; w/o div: our method without using the diversity loss in Eq. 8

	datasets methods	MUTAG	PTC_MR	IMDB-BIN	IMDB-MULTI	REDDIT-BIN
kernel	SP [3]	85.2 ± 2.4	58.2 ± 2.4	55.6 ± 0.2	38.0 ± 0.3	64.1 ± 0.1
	GK [35]	81.7 ± 2.1	57.3 ± 1.4	65.9 ± 1.0	43.9 ± 0.4	77.3 ± 0.2
	WL [34]	80.7 ± 3.0	58.0 ± 0.5	**72.3 ± 3.4**	**47.0 ± 0.5**	68.8 ± 0.4
	DGK [50]	87.4 ± 2.7	60.1 ± 2.6	67.0 ± 0.6	44.6 ± 0.5	**78.0 ± 0.4**
	MLG [22]	**87.9 ± 1.6**	**63.3 ± 1.5**	66.6 ± 0.3	41.2 ± 0.0	–
supervised	GRAPHSAGE [11]	85.1 ± 7.6	63.9 ± 7.7	72.3 ± 5.3	50.9 ± 2.2	–
	GCN [21]	85.6 ± 5.8	64.2 ± 4.3	74.0 ± 3.4	51.9 ± 3.8	50.0 ± 0.0
	GIN-0 [48]	89.4 ± 5.6	64.6 ± 7.0	**75.1 ± 5.1**	**52.3 ± 2.8**	**92.4 ± 2.5**
	GIN-ϵ [48]	89.0 ± 6.0	63.7 ± 8.2	74.3 ± 5.1	52.1 ± 3.6	92.2 ± 2.3
	GAT-ϵ [42]	89.4 ± 6.1	**66.7 ± 5.1**	70.5 ± 2.3	47.8 ± 3.1	85.2 ± 3.3
	PATCHY [26]	**92.6 ± 4.2**	60.0 ± 4.8	71.0 ± 2.2	45.2 ± 2.8	86.3 ± 1.6
unsupervised	RANDOM [6]	83.7 ± 1.5	57.9 ± 1.3	50.7 ± 0.3	34.7 ± 0.2	–
	N2VEC [10]	72.6 ± 10.2	58.6 ± 8.0	–	–	–
	S2VEC [1]	61.1 ± 15.8	60.0 ± 6.4	55.3 ± 1.5	36.7 ± 0.8	71.5 ± 0.4
	INFOGRAPH [37]	89.0 ± 1.1	61.7 ± 1.4	73.0 ± 0.9	49.7 ± 0.5	82.5 ± 1.4
	GRAPHCL [53]	86.8 ± 1.3	–	71.1 ± 0.4	–	**89.5 ± 0.8**
	GCC-MOCO [31]	–	–	73.8 ± 1.1	50.3 ± 0.8	87.6 ± 0.9
	GCC-RAND [31]	–	–	**75.6 ± 0.9**	50.9 ± 0.6	87.8 ± 0.7
	MVGRL [12]	89.7 ± 1.1	62.5 ± 1.7	74.2 ± 0.7	51.2 ± 0.5	84.5 ± 0.6
	MVGRL† [12]	90.1 ± 0.7	62.2 ± 1.1	74.4 ± 0.8	51.2 ± 0.7	85.1 ± 0.4
	JOAO [52]	87.3 ± 1.0	–	70.2 ± 3.1	–	88.1 ± 0.2
	JOAO v2 [52]	87.6 ± 0.8	–	70.8 ± 0.2	–	88.8 ± 0.6
	InfoGCL [47]	91.2 ± 1.3	63.5 ± 1.5	75.1 ± 0.9	51.4 ± 0.8	–
ours	*mo*-mix	91.8 ± 0.4	64.5 ± 0.4	75.1 ± 0.7	52.6 ± 0.5	88.4 ± 0.4
	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
	mp-mix	**92.0 ± 0.5**	**64.6 ± 0.4**	**75.6 ± 0.5**	**52.9 ± 0.4**	88.7 ± 0.4

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 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. **ii) evaluation and results.** We conduct graph edit experiments on [32]. In line with [32], 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 than the negative pair, in a triplet 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 [15] and WL-KERNEL [34]. 2) deep learning-based, i.e., supervised and MVGRL pre-trained. The results in Table 10 show that our method outperforms other GNN methods. We find that with a larger number of nodes, kernel-based methods outperform GNN-based methods. In our analysis, one reason is that kernel-based methods treat each node equally, while GNN-based methods are likely to pay more attention to important nodes in a sub-graph and this may hurt the global performance for GED. This also explains why the AUC score of GNN-based methods is higher than kernel-based methods but has a lower accuracy.

Cluster setup and results. Followed by [12], we evaluate our method under clustering evaluation protocol and cluster the learned representations using the K-Means algorithm for node classification in Table 5. In detail, we set the number of clusters to the number of ground-truth classes and report the average normalized mutual information score (NMI) and adjusted rand score (ARI).

Table 4: Graph classification accuracy w/ different objectives.

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	**92.0 ± 0.5**	**64.6 ± 0.4**	**75.6 ± 0.5**	**52.9 ± 0.4**	**88.7 ± 0.4**

Table 5: Normalized MI (NMI) and adjusted rand index (ARI) on node clustering. The score is calculated by scikit-learn.

	method	Cora		Citeseer		Pubmed	
		NMI	ARI	NMI	ARI	NMI	ARI
unsupervised	VGAE [21]	0.3292	0.2547	0.2605	0.2056	0.3108	0.3018
	MGAE [45]	0.5111	0.4447	0.4122	0.4137	0.2822	0.2483
	ARGA [28]	0.4490	0.3520	0.3500	0.3410	0.2757	0.2910
	ARVGA [28]	0.4500	0.3740	0.2610	0.2450	0.1169	0.0777
	GALA [29]	0.5767	0.5315	0.4411	0.4460	0.3273	0.3214
	MVGRL [12]	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**

Table 6: Graph and node classification by 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 $\mathrm{z} = (\mathrm{z}^a + \mathrm{z}^a)/2$ and $\mathrm{h} = (\mathrm{h}^b + \mathrm{h}^b)/2$. Like ensemble, the aggregated output is more stable than individual output.

Branch	node classification			graph classification			
	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	**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**

4.4 Ablation Study

Comparison of M-Mix-wp, M-Mix-op and binary mix. In Section 3, we have designed an approach named M-Mix-wp, to avoid utilizing prior adjacency matrix. Hence we are able to evaluate this approach in both vision (no adjacency matrix is available) 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 [4] as baselines, and use ResNet-18 [14] as backbone. We set the 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 the site[8]. Table 7 shows the accuracy with different mixing strategies, where multi-sample mixing achieves better results than binary mixing. Note that there's no prior adjacency matrix information to use in vision, thus we only compare our M-Mix-op 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 future directions worth exploring is estimating "adjacency knowledge" across images in vision.

[8] https://github.com/leftthomas/SimCLR

Table 7: Top-1 classification accuracy on node (with MVGRL) and image (with SimCLR) by 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 8: Node classification accuracy. GRACE†: our reproduce – and the variants in bracket is based on GRACE†.

Dataset	Protocol	GRACE	GRACE†	GRACE (GAT)	GRACE+(*i*-mix [23])	GRACE+(*mp*-mix w/o div)	GRACE+(*mp*-mix)
DBLP	Transductive	84.2 ± 0.1	83.8 ± 0.2	84.3 ± 0.2	84.6 ± 0.2	85.2 ± 0.2	**85.6 ± 0.1**
PPI	Inductive	66.2 ± 0.1	66.3 ± 0.2	66.3 ± 0.1	66.4 ± 0.3	67.3 ± 0.2	**67.8 ± 0.1**

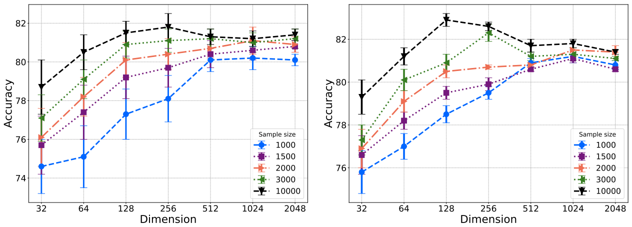

Figure 2: Node classification accuracy (%) on Pubmed dataset with different sample sizes and different hidden dimensions. (Left: w/o div loss; Right: w/ div loss)

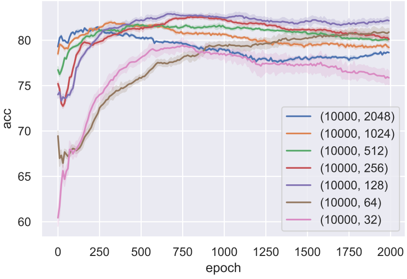

Figure 3: Accuracy (%) and standard variance with different batch / hidden pairs over training epochs. For instance, $(10000, 2048)$ **means training with** $N = 10000$, $\mathcal{F}' = 2048$.

Comparison with GAT. As discussed in Section 3, *mp*-mix without diversity loss is somewhat similar to GAT [41], while *mo*-mix and *mp*-mix are completely different from GAT in terms of both methodology and technology. Here, we further design experiments to clarify the difference. We evaluate *mp*-mix with the baseline GRACE [61] on DBLP and PPI benchmarks. As shown in Table 8, 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 multi-sample mixing methods, *mp*-mix outperforms baseline GRACE 1.4% and 1.6% accuracy on DBLP and PPI, respectively.

Mixing weights. To further explore the effect of different mixing weights, we design a baseline, i.e., randomly generating mixing weights from Gaussian distribution and regularizing them with Softmax function. Table 9 shows the accuracy of different mixing weights. Although randomly generated value can get a little improvement over baseline MVGRL, it suffers from instability (high variance). Then, by replacing the random mix with the proposed *mo*-mix and *mp*-mix with diversity loss, the performance is improved by a notable margin in both accuracy and stability.

Output dimension and sample size. Contrastive learning usually requires a large number of negative pairs. To evaluate the robustness of our method w.r.t the number of negative samples (i.e., subgraph sample size), we try different combinations of sample size and output dimension. Fig. 2 shows the result on Pubmed. Different from the empirical conclusion in vision [4, 13] (large output dimension leads to better results (saturate at 4096)), we find that a proper combination of output dimension and the sample size is important in graph contrastive learning, which is in line with Theorem 3.2. Empirically, a smaller output 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 output dimension is too small (i.e., 32), the information can not be fully represented. Thus, the best accuracy is obtained when the output dimension is set as 256. On the other hand, we can not ignore the advantage of large output 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) compared with low output dimension.

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 4 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 report 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 6 shows, both two branches achieve remarkable results.

Detail of ablation experiments on different sample sizes and output dimensions. We test our method under different sample sizes and output dimensions. We set the epoch as 2000 and record the accuracy every 10 epochs. We find with a higher output dimension, the model's stability is increasing, meanwhile,

Table 9: Accuracy (%) comparison. The first line MVGRL means taking normalized adjacency and diffusion matrix as input to perform contrasting and the loss is the same as [12]. The remaining lines use objectives of this paper $\mathcal{J}_{node}$ in Eq. 10 and $\mathcal{J}_{graph}$ in Eq. 11. MVGRL+(random mix) means the designed baseline. We fine-tune hyper-parameters for 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**

Table 10: Accuracy (top) and AUC (bottom) on COIL-DEL for the task of graph edit distance. HIST-KERNEL means vertex edge hist kernel; n is the number of nodes in sampled subgraph. We use the implementation of the two kernel methods, i.e., HIST-KERNEL [15] and WL-KERNEL [34] provided by [36] to produce the results.

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

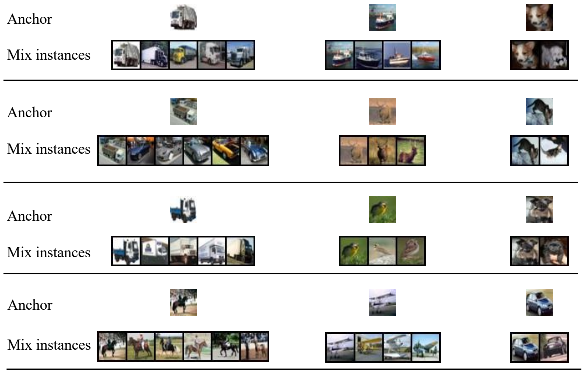

Figure 4: Visualization of multi-sample mixing on CIFAR-10, where anchor means the input query image and mix samples mean the selected mixing instances by M-Mix.

the convergence rate is faster than the low dimension. When set output 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 output dimension, initialed embedding has remarkable representation ability. We guess it is because, with a larger output dimension, the initialized node embedding vector is closer to the optimal solution (orthogonal in hypersphere).

Visualization on Vision Datasets. Our M-Mix can adaptively select which samples should be mixed and gives them different mixing weights. Here we randomly select anchor samples in CIFAR-10 and CIFAR-100, and visualize the mixing samples selected by M-Mix in Fig. 4 and Fig. 5, respectively. The mined samples have similar semantic information with anchor images, which greatly improves the difficulty of discrimination after mixing.

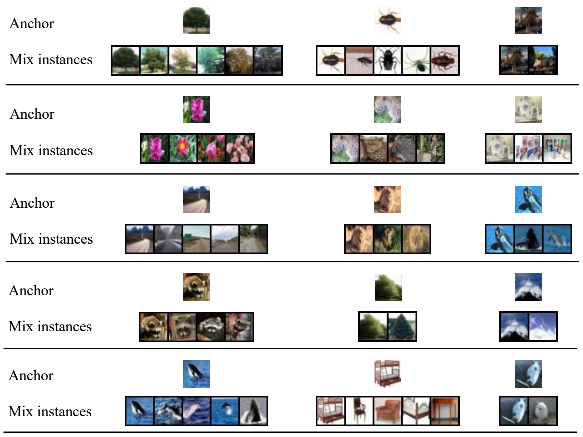

Figure 5: Visualization of samples mixing on CIFAR-100.

5 CONCLUSION

In this paper, we have proposed a new *mp*-mix to mine hard negatives for contrastive learning, which mixes multi-sample and assigns different mixing weights dynamically. To our best knowledge, this is the first attempt at mixing multiple samples other than the two common practices of mixing two samples, and also the first work for learning mixing weights dynamically. Then, we theoretically analyze why the proposed strategy of assigning mixing weight is better than others. We further provide theoretical analysis on the relation of output dimension and sample size, and show how to estimate their suitable values. 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 two downstream experiments on clustering and graph edit distance. The results imply our method can stabilize the model with high transferability. Seeing its exhibited good generalization ability, our method can be potentially applied in other domains, such as NLP and cross-modal pretraining, which we leave for future work.

REFERENCES

[1] Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. 2018. Sub2vec: Feature learning for subgraphs. In *PAKDD*.

[2] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR* (2006).

[3] Karsten M Borgwardt and Hans-Peter Kriegel. 2005. Shortest-path kernels on graphs. In *ICDM*.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *NeurIPS* (2016).

[6] Thomas Gärtner, Peter Flach, and Stefan Wrobel. 2003. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*.

[7] Lise Getoor. 2005. Link-based classification. In *Advanced methods for knowledge discovery from complex data*.

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

[9] 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. 2020. Bootstrap your own latent: A new approach to self-supervised learning. *NeurIPS* (2020).

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*.

[11] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

[12] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *ICML*.

[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

[15] Shohei Hido and Hisashi Kashima. 2009. A linear-time graph kernel. In *ICDM*.

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

[17] Qianjiang Hu, Xiao Wang, Wei Hu, and Guo-Jun Qi. 2020. AdCo: Adversarial Contrast for Efficient Learning of Unsupervised Representations from Self-Trained Negative Adversaries. *arXiv preprint* (2020).

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

[19] Sungnyun Kim, Gihun Lee, Sangmin Bae, and Se-Young Yun. 2020. MixCo: Mix-up Contrastive Learning for Visual Representation. *arXiv preprint arXiv:2010.06300* (2020).

[20] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[21] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[22] Risi Kondor and Horace Pan. 2016. The multiscale laplacian graph kernel. *arXiv preprint* (2016).

[23] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. 2020. *i*-Mix: A Domain-Agnostic Strategy for Contrastive Representation Learning. In *ICLR*.

[24] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. 2019. Graph matching networks for learning the similarity of graph structured objects. In *ICML*.

[25] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*.

[26] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *ICML*.

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

[28] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding. In *IJCAI*.

[29] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. 2019. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *ICCV*.

[30] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*.

[31] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *KDD*.

[32] Kaspar Riesen and Horst Bunke. 2008. 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)*.

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

[34] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-lehman graph kernels. *JMLR* (2011).

[35] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. 2009. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*.

[36] Mahito Sugiyama, M Elisabetta Ghisu, Felipe Llinares-López, and Karsten Borgwardt. 2018. graphkernels: R and Python packages for graph comparison. *Bioinformatics* (2018).

[37] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. 2019. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.

[38] Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020. Contrastive multiview coding. In *ECCV*.

[39] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. 2018. Verse: Versatile graph embeddings from similarity measures. In *WWW*.

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

[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[42] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *stat* (2018).

[43] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax.. In *ICLR*.

[44] Vikas Verma, Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc Le. 2021. Towards domain-agnostic contrastive learning. In *ICML*.

[45] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. 2017. Mgae: Marginalized graph autoencoder for graph clustering. In *CIKM*.

[46] Xiangyu Wang, Chenlei Leng, and David B Dunson. 2015. On the consistency theory of high dimensional variable screening. In *NeurIPS*.

[47] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. 2021. Infogcl: Information-aware graph contrastive learning. *NeurIPS* (2021).

[48] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In *ICLR*.

[49] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *ICML*.

[50] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *KDD*.

[51] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*.

[52] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph Contrastive Learning Automated. *ICML* (2021).

[53] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *NeurIPS* (2020).

[54] Zhiping Zeng, Anthony KH Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. 2009. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment* (2009).

[55] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond Empirical Risk Minimization. In *ICLR*.

[56] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and S Yu Philip. 2021. From canonical correlation analysis to self-supervised graph neural networks. In *NeurIPS*.

[57] Shaofeng Zhang, Lyn Qiu, Feng Zhu, Junchi Yan, Hengrui Zhang, Rui Zhao, Hongyang Li, and Xiaokang Yang. 2022. Align Representations with Base: A New Approach to Self-Supervised Learning. In *CVPR*.

[58] Shaofeng Zhang, Feng Zhu, Junchi Yan, Rui Zhao, and Xiaokang Yang. 2021. Zero-CL: Instance and Feature decorrelation for negative-free symmetric contrastive learning. In *ICLR*.

[59] Qiang Zhou, Chaohui Yu, Zhibin Wang, Qi Qian, and Hao Li. 2021. Instant-Teaching: An End-to-End Semi-Supervised Object Detection Framework. In *CVPR*.

[60] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*.

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

[62] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph Contrastive Learning with Adaptive Augmentation. In *WWW*.