Information Sciences 657 (2024) 119952

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

# Breaking the curse of dimensional collapse in graph contrastive learning: A whitening perspective

Yang Tao [a], Kai Guo [a], Yizhen Zheng [b], Shirui Pan [c], Xiaofeng Cao [a,*], Yi Chang [a]

[a] School of Artificial Intelligence, Jilin University, Changchun, Jilin Province, 130012, China
[b] Department of Data Science & AI, Monash University, Melbourne, Victoria, 3168, Australia
[c] School of Information and Communication Technology, Griffith University, Brisbane, Queensland, 4222, Australia

ARTICLE INFO

*Keywords:*
Graph contrastive learning
Dimensional collapse
Whitening

ABSTRACT

Dimensional collapse in graph contrastive learning (GCL) confines node embeddings to their lower-dimensional subspace, diminishing their distinguishability. However, the causes and solutions of this curse remain relatively underexplored. In statistics, whitening presents a powerful tool to eliminate correlations among multiple variables. This motivates us to relieve the dimensional collapse of GCL from a whitening perspective. In this paper, we propose an intuitive analysis suggesting that high similarity scores of node embeddings may cause dimensional collapse, providing more evidence for its presence. Considering the success of whitening in statistics, we introduce a new plug-and-play module called the Whitening Graph Contrastive Learning (WGCL) to address the dimensional collapse issue in existing GCL methods. WGCL plugin standardises the covariance matrices of dimensions, eliminating correlations among node embeddings' dimensions. Additionally, we enhance the conventional GCL training objective by introducing a mutual information maximisation loss between input features and node embeddings to maintain information capacity. Our experiments demonstrate that WGCL effectively addresses dimensional collapse, leading to an average improvement of 0.93% (up to 2.0%) in classification accuracy across three GCL backbones on nine widely-used datasets. The code to reproduce the experiments is available at https://github.com/acboyty/WGCL.

## 1. Introduction

Graph representation learning (GRL) is currently a prevailing method for analysing graph-structured data, with graph neural networks (GNNs) being particularly notable for their exceptional performance [1,2]. Many existing GNN models rely on supervised learning, which is hindered by limitations in the amount and cost of labelled data. As an effective approach to reduce the need for extensive annotations, contrastive learning (CL) has shown remarkable progress in various fields, including visual representation learning [3–5] and natural language processing [6,7], by producing representations that are robust to augmentations [8]. Therefore, GNN community has transferred the success of CL methods to the field of graphs, leading to a new research direction named graph contrastive learning (GCL).

* Corresponding author.
*E-mail addresses:* taoyang21@mails.jlu.edu.cn (Y. Tao), guokai20@mails.jlu.edu.cn (K. Guo), yizhen.zheng1@monash.edu (Y. Zheng), s.pan@griffith.edu.au (S. Pan), xiaofengcao@jlu.edu.cn (X. Cao), yichang@jlu.edu.cn (Y. Chang).

https://doi.org/10.1016/j.ins.2023.119952
Received 11 August 2023; Received in revised form 13 October 2023; Accepted 25 November 2023

Available online 29 November 2023

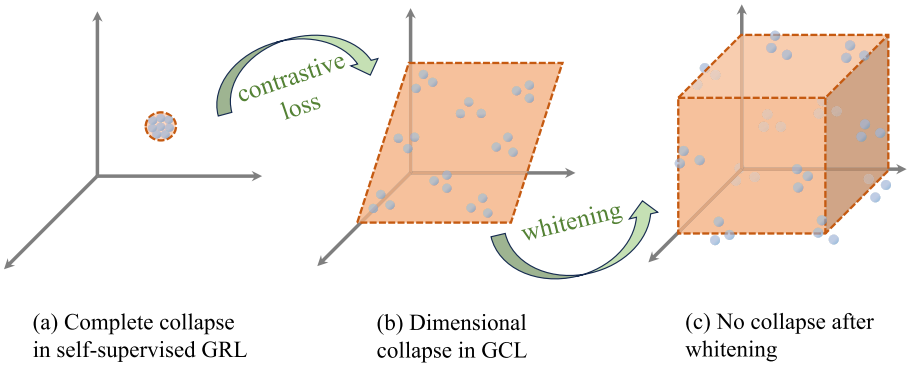



**Fig. 1.** Two collapse scenarios of embeddings generated by self-supervised GRL and our whitened one in a 3-dimensional (3D) space. (a) A complete collapse occurs when self-supervised GRL focus solely on minimising the distance between positive pairs, converging to the trivial solution of mapping all nodes to the same position in the embedding space. (b) Dimensional collapse is a common defect in most existing GCL methods, referring to the phenomenon where the rank of the embeddings $d'$ is smaller than the dimension of embedding space $d$, resulting in embeddings lying only on a 2D plane subspace in a 3D space (In section 4, we explore this phenomenon and provide the theoretical explanation). (c) GCL methods integrated with our plug-and-play module WGCL, produce embeddings that cover the whole 3D space, leading to improved expressivity and more comprehensive information encoding.

In GCL, standard approaches use an instance (node) discrimination task where augmented versions of the same instance are "positive pairs", while different instances are "negative pairs". "Positive" denotes that different views of the same instance retain the same semantic information, while "negative" signifies that various views of different instances likely have distinct semantics. Under this framework, several typical GCL works have been proposed. For example, the seminal work Deep Graph InfoMax (DGI) [9] tackles the problem of unsupervised graph learning by augmenting the sampled graph with randomly shuffled node features to construct positive instances. Then, an InfoMax-based objective function is proposed to maximise the mutual information between node embeddings and a global summary embedding. Building on DGI, a series of GCL methods [10–15] further enforce embeddings of positive pairs to be close and embeddings of negative pairs to be distant. Moreover, GCA [16] proposes an adaptive graph augmentation approach to improve the performance of GCL from the perspective of input data.

Despite recent remarkable progress, GCL is known to suffer from the curse of dimensional collapse [17,18], where the node embeddings become confined to their lower-dimensional subspace. For example, in three-dimensional (3D) space, ideal embeddings should adequately span the entire 3D space. However, due to the curse of dimensionality collapse, these embeddings may be restricted to a 2D space as illustrated in Fig. 1(b). DirectCLR [17] sheds light on the strong augmentation in vanilla CL that leads to dimensional collapse. Then nmrGCL [18] transferred this insight to GCL, which proposes two reasons for dimensional collapse: the smoothing effect of graph pooling layers and the implicit regularisation of graph convolution layers. The underutilisation of dimensions weakens the distinguishability of node embeddings, leading to sub-optimal model performance in downstream tasks. Unfortunately, despite the growing interest in GCL, the causes of the dimensional collapse and the corresponding techniques to effectively address this issue remain relatively underexplored.

In this paper, we devote a great effort to explore the reason why the conventional GCL suffers from dimensional collapse and sub-optimisation. Through theoretical analysis and empirical observation, we find that optimisation objective of GCL encourages the model to focus on maximising the magnitude of the similarity of node embeddings, rather than considering the overall structure of the latent space, leading to dimensional collapse. In statistics, whitening presents a powerful decomposition tool that can eliminate correlations among multiple variables. This motivates us to relieve the dimensional collapse of GCL from a whitening perspective. We propose a novel and simple plug-and-play module named Whitening Graph Contrastive Learning, WGCL for brevity, to effectively decorrelate the dimensions of the generated node embeddings. The key idea is to apply whitening techniques to standardise the covariance matrices of embedding dimensions, inspired by Decorrelated Batch Normalisation [19], a technique recently used in visual contrastive learning [8]. Based on this technique, we consider the unbalanced dimensions of graphs and propose node-shuffled whitening, producing node embeddings covering the whole embedding space (Fig. 1(c)). Furthermore, given that dimensional collapse can result in reduced information, we introduce a loss function that maximises mutual information between inputs and outputs, thus enhancing embedding information capacity. The proposed WGCL plugin is evaluated with 3 GCL backbones against 16 baselines on 9 widely-used datasets, and the results demonstrate its effectiveness. **The contributions of this paper include:**

- **Novel observation.** We conducted an intuitive observation from a new perspective, i.e., the magnitude of similarity score of embeddings. On commonly used graph datasets, we confirm the theoretical inference and the existence of dimensional collapse in GCL.
- **Plug-and-play module.** From whitening in statistical learning, we introduce a novel and simple plug-and-play module called Whitening Graph Contrastive Learning (WGCL) plugin to address the dimensional collapse issue for existing GCL methods. Our proposed plugin encourages the encoder to produce non-collapsed embeddings by utilising whitening techniques.
- **Exceptional performance.** The proposed WGCL module has been thoroughly tested on 9 widely-used datasets with 3 most competitive GCL backbones. Experimental results show its effectiveness in improving existing GCL methods, with an average accuracy enhancement of 0.93% (up to 2.0%), and it achieves new state-of-the-art results against 16 baselines.

**Table 1**
Summary of the main symbols and notations.

| Symbols | Description |
|---|---|
| $\mathcal{G}$ | A graph. |
| $\mathcal{V}$ | The set of nodes in a graph. |
| $\mathcal{E}$ | The set of edges in a graph. |
| $\mathbf{X} \in \mathbb{R}^{N \times d}$ | Input feature matrix. |
| $\mathbf{A} \in \{0,1\}^{N \times N}$ | Adjacency matrix of graph. |
| $\mathbf{X}_*$, $\mathbf{A}_*$ | Node feature and adjacency matrix of corresponding graph view ($*$ denotes the view number, i.e., 1 or 2). |
| $\mathbf{H} \in \mathbb{R}^{N \times d'}$ | Embedding matrix in final layer. |
| $\mathbf{H}_l \in \mathbb{R}^{N \times d_l}$ | Embedding matrix in layer $l$. |
| $\mathbf{H}'_l$ | Embedding matrix after whitening operation in layer $l$. |
| $\mathbf{H}^{(*)}$, $\mathbf{H}_l^{(*)}$ | Embedding matrix of corresponding graph view ($*$ denotes the view number, i.e., 1 or 2). |
| $\mathbf{I}$ | Identity matrix. |
| $\|\mathbf{s}\|$, $\hat{\mathbf{s}}$ | Magnitude and direction of the similarity vector $\mathbf{s}$. |
| $L$ | The total layer numbers of the GNN model. |
| $\alpha$ | Parameter to tune the trade-off between InfoNCE loss and whitening-enhancing loss. |
| $\circ$ | Function composition. |

## 2. Related works

### 2.1. Graph contrastive learning (GCL)

Graph representation learning [1,2] finds applications in diverse fields, such as social network analysis [20], recommendation systems [21], bioinformatics [22], and hybrid clustering [23]. Contrastive learning is a self-supervised representation learning approach that has gained popularity in computer vision [24,4] and natural language processing [25,26], which has inspired the development of a series of graph contrastive learning methods. Deep Graph InfoMax (DGI) [9] was the first to apply the InfoMax principle to graph representation learning by maximising the mutual information between patch-level and global-level representation of a graph. GMI [10] jointly maximises feature MI and edge MI individually, without augmentation. MVGRL [11] generates two augmented graph views via graph diffusion and subgraph sampling. CuCo [27] further utilises curriculum learning to select negative samples. AD-GCL [28] optimises adversarial graph augmentation to prevent learning redundant information. JOAO [29] adaptively selects the augmentation for a specific dataset.

### 2.2. Collapse in self-supervised learning

Self-supervised contrastive learning methods have been shown to suffer from a collapse problem, where the learned embeddings become a constant vector. To address this issue, MoCo [4] and SimCLR [24] repulse negative pairs in the optimisation objective, while BYOL [30] proposes a momentum encoder, predictor, and stop gradient operator to avoid collapsed solutions. SimSiam [31] simplifies the BYOL approach by removing the momentum encoder and demonstrating that the remaining stop-gradient mechanism is key to preventing collapse in self-supervised learning. Recent studies [8,17] in the vision field have also pointed out the dimensional collapse issue, where strong augmentation and weight matrix alignment can cause the learned embeddings to collapse to a lower-dimensional space.

## 3. Preliminaries

In this section, we present the overall framework of graph contrastive learning (GCL) and introduce the key notations used in this paper. Main symbols (including subsequent sections) could be easily referred to in Table 1.

### 3.1. Setup

Specifically, we denote a graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of $N$ nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges describing the relations between nodes, and $\mathbf{X}$ is the node feature matrix of size $N \times d$ ($d$ denotes the feature dimension). The graph structure can also be represented by an adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$, where $\mathbf{A}_{ij} = 1$ indicates the existence of an edge between nodes $v_i$ and $v_j$, and $\mathbf{A}_{ij} = 0$ otherwise. Therefore, a graph can be denoted as $\mathcal{G} = (\mathbf{X}, \mathbf{A})$.

GCL aims to train a graph neural network (GNN) encoder $f(\cdot)$ that takes both the node feature and graph structure as input and produces node embeddings $\mathbf{H}$ with a feature dimension of $d'$. Specifically, $\mathbf{H}$ is obtained as $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_N]^T = f(\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{N \times d'}$. These node embeddings can be leveraged in downstream tasks such as node classification and anomaly detection. Typically the encoder usually consists of $L$ GNN layers, where each layer takes the output of the previous layer as the input, i.e., $f = f_L \circ f_{L-1} \circ \ldots \circ f_1$. Each GNN layer $f_l$ updates the representations of all nodes by propagating and transforming representations of their neighbours, i.e., $\mathbf{H}_l = [\mathbf{h}_{l,1}, \ldots, \mathbf{h}_{l,N}]^T = f_l(\mathbf{H}_{l-1}, \mathbf{A})$. Note that $\mathbf{H}_0 = \mathbf{X}, \mathbf{H}_L = \mathbf{H}$.

### 3.2. Graph contrastive learning framework

In a typical graph contrastive learning (GCL) framework, the objective is to minimise the disagreement between node embeddings of positive instances while maximising the distinguishability between those of negative instances. To achieve this, given a graph $\mathcal{G}$, we select two augmentation functions $t_1$ and $t_2$ from a predefined graph augmentation set $\mathcal{T}$ including:

- *Graph perturbation:* Modifying the graph structure by adding, deleting, or rewiring edges to create variations of the original graph. [32]
- *Node feature perturbation:* Modifying the node features, such as adding noise or changing their values, to create variations of the original graph. [33]
- *Subgraph extraction:* Extracting subgraphs from the original graph and using them as separate training samples. [9]
- *Graph augmentation through generative models:* Using generative models such as variational autoencoder (VAE) and graph convolutional networks (GCN) to generate new graph structures or features. [34]

These augmentation functions are used to generate two augmented graph views, denoted as $\mathcal{G}_1 = t_1(\mathcal{G})$ and $\mathcal{G}_2 = t_2(\mathcal{G})$. Next, the GNN encoder is utilised to produce node embeddings of the two different views, denoted as $\mathbf{H}^{(1)} = f(\mathbf{X}_1, \mathbf{A}_1)$ and $\mathbf{H}^{(2)} = f(\mathbf{X}_2, \mathbf{A}_2)$, where $\mathbf{X}_*$ and $\mathbf{A}_*$ are node feature and adjacency matrices of corresponding $\mathcal{G}_*$ ($*$ denotes the view number, i.e., 1 or 2).

Then, we proceed to adopt an instance-discrimination task, which aims to simultaneously pull positive pairs closer and push negative pairs further apart. For each node $v_i$, we denote its embedding in one view as $\mathbf{h}_i^{(1)}$. The embedding $\mathbf{h}_i^{(2)}$ in the other view forms the positive instance, while the embeddings of other nodes in the two views are considered as negative instances. To adapt the typical InfoNCE loss [35] to the GCL setting, we define the objective for each augmented node $v_i$ in one view as follows:

$$\mathcal{L}_i^{(a)} = -\log \frac{e^{s(\mathbf{h}_i^{(a)}, \mathbf{h}_i^{(b)})}}{\underbrace{e^{s(\mathbf{h}_i^{(a)}, \mathbf{h}_i^{(b)})}}_{\text{positive pair}} + \underbrace{\sum_{k \neq i} e^{s(\mathbf{h}_i^{(a)}, \mathbf{h}_k^{(b)})}}_{\text{inter-view negative pairs}} + \underbrace{\sum_{k \neq i} e^{s(\mathbf{h}_i^{(a)}, \mathbf{h}_k^{(a)})}}_{\text{intra-view negative pairs}}}, \tag{1}$$

where $s(\cdot, \cdot)$ denotes similarity score such as linear similarity, cosine correlation, etc., the possible value combinations for $a$ and $b$ are $a = 1, b = 2$ or $a = 2, b = 1$. The overall loss is $\mathcal{L}_{\text{NCE}} = \frac{1}{2N} \sum_{i \in \{1, \cdots, N\}} (\mathcal{L}_i^{(1)} + \mathcal{L}_i^{(2)})$.

## 4. Exploring dimensional collapse in GCL

In the following section, we aim to explore why the conventional graph contrastive learning (GCL) framework, trained with the widely used InfoNCE loss, often yields lower-dimensional embeddings. Our analysis inspired by [36] indicates that the high similarity of node embeddings produced by this framework may be one of the contributing factors. We have conducted an empirical study to validate this finding.

### 4.1. Theoretical insight

For the sake of clarity, we will first focus on $\mathcal{L}_i^{(1)}$ defined in Eq. (1). To this end, we introduce the similarity vector $\mathbf{s} = [s_1, \ldots, s_{2N-1}] \in \mathbb{R}^{2N-1}$, which contains the similarity scores $s(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)})$, $\{s(\mathbf{h}_i^{(1)}, \mathbf{h}_k^{(2)})\}_{k \neq i}$ and $\{s(\mathbf{h}_i^{(1)}, \mathbf{h}_k^{(1)})\}_{k \neq i}$ sequentially.

**Definition 1** *(Magnitude and direction of* $\mathbf{s}$*). Given the similarity vector* $\mathbf{s}$*, we could decompose it as follows:*

$$\mathbf{s} = \|\mathbf{s}\| \cdot \hat{\mathbf{s}}, \tag{2}$$

*where* $\|\mathbf{s}\| = (s_1^2 + s_2^2 + \cdots + s_{2N-1}^2)^{1/2}$ *is the* $L^2$*-norm of the similarity vector* $\mathbf{s}$*, and* $\hat{\mathbf{s}} = [\hat{s}_1, \ldots, \hat{s}_{2N-1}]$ *is the unit vector in the same direction as* $\mathbf{s}$*. To put it another way,* $\|\mathbf{s}\|$ *and* $\hat{\mathbf{s}}$ *correspond to the magnitude and direction of the similarity vector* $\mathbf{s}$*, respectively.*

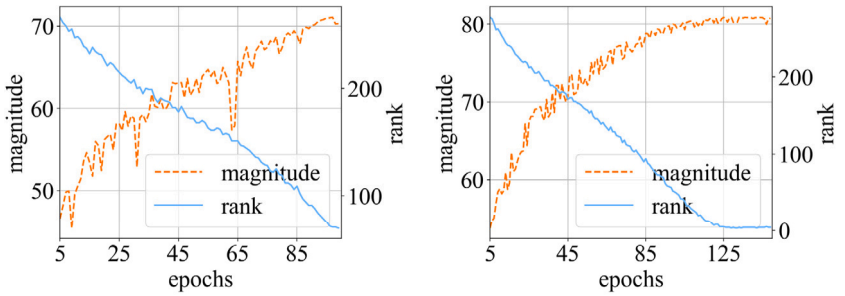
To examine the impacts of both magnitude and direction on the training objective, we derive the formula based on Eq. (1), (2):

$$\mathcal{L}_i^{(1)} = -\log \frac{e^{s_1}}{\sum_{i=1}^{2N-1} e^{s_i}} = -\log \frac{e^{\|\mathbf{s}\| \cdot \hat{s}_1}}{\sum_{i=1}^{2N-1} e^{\|\mathbf{s}\| \cdot \hat{s}_i}}, \tag{3}$$

where we could observe that the training loss is determined by $\|\mathbf{s}\|$ and $\hat{\mathbf{s}}$. By fixing the direction $\hat{\mathbf{s}}$ to a constant value, we can focus on the effects of $\|\mathbf{s}\|$ on the training objective.

**Proposition 1.** *Given that* $s_1$ *is the maximum value of* $s_i$*, we could obtain that* $\arg\max_i (\hat{s}_i) = 1$*. Let* $t = \|\mathbf{s}\| - 1$*, then we have,*

$$\frac{e^{\|\mathbf{s}\| \hat{s}_1}}{\sum_{i=1}^{2N-1} e^{\|\mathbf{s}\| \hat{s}_i}} = \frac{e^{(1+t)\hat{s}_1}}{\sum_{i=1}^{2N-1} e^{(1+t)\hat{s}_i}} = \frac{e^{\hat{s}_1}}{\sum_{i=1}^{2N-1} e^{\hat{s}_i + t(\hat{s}_i - \hat{s}_1)}}.$$




(a) Cora (b) CiteSeer

**Fig. 2.** The magnitudes of the similarity vectors with respect to $\mathcal{L}_i^{(1)}$ and the numerical rank are compared across different epochs, which validates the point of Remark 1. The model was trained on the Cora and CiteSeer datasets using a vanilla 2-layer GCN with GRACE [13] contrastive learning framework.

*For any* $i \in [1, 2, \ldots, 2N-1]$*, we have,* $\hat{s}_i - \hat{s}_1 \leq 0$*. Then for* $\|\mathbf{s}\| \geq 1$*,*

$$\frac{e^{\|\mathbf{s}\|\hat{s}_1}}{\sum_{i=1}^{2N-1} e^{\|\mathbf{s}\|\hat{s}_i}} \geq \frac{e^{\hat{s}_1}}{\sum_{i=1}^{2N-1} e^{\hat{s}_i}}.$$

*It can be observed that as the magnitude of the vector* $\mathbf{s}$ *increases, the value of the loss term* $\mathcal{L}_i^{(1)}$ *decreases.*

We then proceed to our main insight to uncover the underlying reason behind dimensional collapse. After optimising $\hat{\mathbf{s}}$ to a certain level, i.e., $s_1 = max_i s_i$, it can be inferred from Proposition 1 that the magnitude $\|\mathbf{s}\|$ of similarity score will be further amplified to optimise the training objective. Recalling the definition of $\mathbf{s}$, this indicates that the two embeddings in parentheses $(\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)})$, as well as $\{(\mathbf{h}_i^{(1)}, \mathbf{h}_k^{(2)})\}_{k \neq i}$ and $\{(\mathbf{h}_i^{(1)}, \mathbf{h}_k^{(1)})\}_{k \neq i}$, may exhibit a high similarity. This conclusion can be readily generalised to $\mathcal{L}_i^{(2)}$ in Eq. (1), suggesting that a strong correlation exists among the $2N$ embeddings $\{\mathbf{h}_i^{(1)}, \mathbf{h}_i^{(2)}\}_{i=1}^N$. When this correlation is further generalised to the most common linear correlation, it becomes obvious that $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$ can be considered low row rank. As we all know, the column rank of one matrix is equal to the row rank, thus $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$ could be considered low column rank. This indicates that there is a strong correlation between the various dimensions. Though the pushing-away strategy utilised by GCL avoids complete collapse (shown in Fig. 1(a)), the low-column-rank property accentuates the occurrence of dimensional collapse.

**Remark 1.** *Optimising the InfoNCE loss (i.e., Eq. (1)) leads to an increase in the similarity score among node embeddings, which in turn leads to the occurrence of dimensional collapse.*

### 4.2. Empirical study

In this section, we conduct an empirical study to corroborate the theoretical insights presented in the previous section. We design an experiment to explore the temporal progression of the magnitudes of the similarity vector $\mathbf{s}$ related to $\mathcal{L}_i^{(1)}$ and the numerical rank of the embedding matrix during the training process. The results of this experiment can be observed in Fig. 2.

From the figure we can observe that, with increasing training epochs, the magnitude of $\mathbf{s}$ grows while the rank of embedding matrix decreases. This indicates that conventional Infomax training loss of GCL enhances the similarity between node embeddings, which manifests in the form of higher norms of similarity vector $\mathbf{s}$. However, this pursuit of similarity leads to a reduction in the rank of the embedding matrix, resulting in dimensional collapse, that can lead to inferior performance in downstream tasks. A direct approach to tackle this issue could involve applying regularisation on $\mathbf{s}$ to limit its magnitude. Nonetheless, this method oversimplifies the problem and may degrade the model performance. Thus, we introduce a novel approach that specifically addresses this problem.

## 5. Whitening graph contrastive learning

In this section, we present Whitening GCL (`WGCL`) plugin (Fig. 3) to address dimensional collapse in detail. We propose node-shuffled whitening as an extension to ZCA whitening, which takes into account the unique characteristics of graph structured data. To further enhance the whitening effect, we propose a well-designed loss function that maximises the mutual information between input features and node embeddings, which better decorrelate feature dimensions.

### 5.1. Whitening

Whitening is a widely used processing technique in various fields, including image processing, signal processing, etc., to improve the quality of embeddings by decorrelating features and equalising their variances. Its primary objective is to increase the independence of data by reducing redundant information and correlation, thereby improving the performance of classifiers.

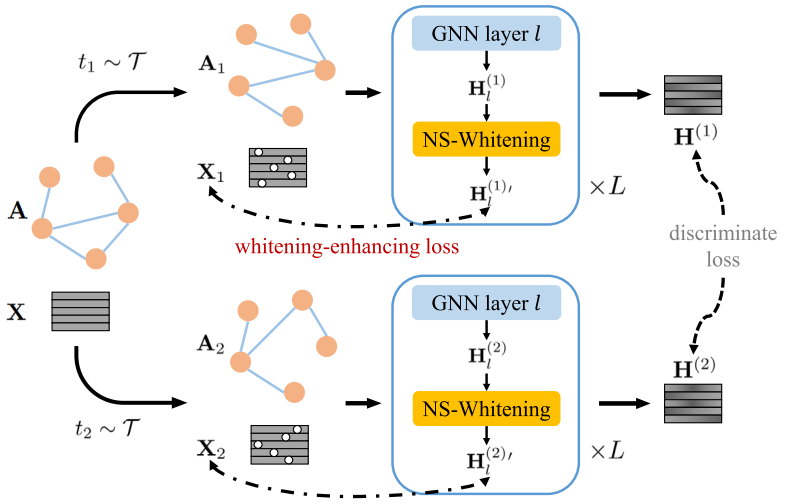



**Fig. 3.** Our proposed Whitening Graph Contrastive Learning (WGCL) plugin improves upon traditional GCL in two ways. First, we incorporate a node-shuffled whitening module after each GNN layer to decorrelate the dimensions of the node embeddings. Second, to further enhance the effect of whitening, we introduce whitening-enhancing loss that maximises the mutual information between the input features and each layer's embeddings. As a result, our WGCL plugin decorrelates the dimensions of node embeddings, leading to the improved downstream task performance.

Considering the inherent speed advantage of non-parametric methods over DNNs, our primary focus lies on traditional PCA/ZCA whitening rather than non-linear methods. As depicted in recent studies [37,19], the most classic PCA whitening approach is proven to be unsuitable for NNs. PCA whitening suffers from stochastic axis swapping during NN training. This phenomenon causes it to behave similarly to random guessing. Hence, in this paper, we adopt ZCA whitening and its definition is provided below.

**Definition 2** *(ZCA whitening). The ZCA whitening algorithm operates on a $d$-dimensional embedding matrix $\mathbf{H} = [\mathbf{h}_1, \dots, \mathbf{h}_N]^T \in \mathbb{R}^{N\times d}$ and produces a matrix $\mathbf{H}' = [\mathbf{h}'_1, \dots, \mathbf{h}'_N]^T \in \mathbb{R}^{N\times d}$ as output. The algorithm performs the following steps to compute $\mathbf{H}'$:*

$$\mathbf{H}' = (\mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^T\hat{\mathbf{H}}^T)^T,$$

*where the matrix $\hat{\mathbf{H}}$ is obtained by normalising the columns of $\mathbf{H}$ to zero-mean. (i.e. $\hat{\mathbf{h}}_{i,j} = \mathbf{h}_{i,j} - \frac{1}{N}\sum_{k=1}^{N}\mathbf{h}_{k,j}$), $\mathbf{\Lambda} \in \mathbb{R}^{d\times d}$ is a diagonal matrix filled with the eigenvalues of $\mathbf{\Sigma} = \hat{\mathbf{H}}^T\hat{\mathbf{H}}$ and $\mathbf{Q} \in \mathbb{R}^{d\times d}$ is the corresponding orthonormal eigenvectors (i.e. $\mathbf{\Sigma} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$). ZCA assumes $\mathbf{\Sigma} = \hat{\mathbf{H}}^T\hat{\mathbf{H}} \in \mathbb{R}^{d\times d}$ is full-rank.*

The columns of the ZCA's output $\mathbf{H}'$ are zero-mean, and therefore the corresponding covariance matrix is

$$\begin{aligned}\mathbf{H}'^T\mathbf{H}' &= \mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^T\hat{\mathbf{H}}\hat{\mathbf{H}}^T\mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^T \\ &= \mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^T\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^T \\ &= \mathbf{Q}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{\Lambda}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{Q}^T = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}.\end{aligned}$$

We can see that after ZCA whitening, the product of vectors between different dimensions is zero, which means that the Pearson correlation between dimensions has been removed. Given that dimensional collapse refers to the excessive correlation among dimensions, **ZCA whitening is proven to be a highly effective solution for mitigating this issue.**

One crucial problem of ZCA whitening is that calculating the whitening matrix $\mathbf{\Lambda}^{-\frac{1}{2}}$ requires eigen-decomposition or SVD, which heavily constrains its practical applications. Thus we use Newton's iteration methods [38] to **estimate** whitening matrix $\mathbf{\Lambda}^{-\frac{1}{2}}$, which avoids executing eigen-decomposition or SVD following [39].

### 5.2. Node-shuffled whitening

In this section, we present node-shuffled whitening (NSW), a whitening method specifically tailored for GNN frameworks.

Though whitening could alleviate the pattern of dimensional collapse which commonly affects the performance of GCL, it cannot be directly applied in GNN frameworks. This is because GNN frameworks typically input all nodes into the network at one iteration, which causes the dimension of $\mathbf{H}_l$ to be unbalanced, namely $N \gg d$. Thus, feature dimensions are naturally less correlated, reducing the difficulty of whitening and finally weakens the learning ability of GCL. To reveal the potential of worse model representation ability, we propose a variant of whitening with one easy add-on, namely node-shuffled whitening (NSW), which divide $\mathbf{H}_l$ into several sub-matrices along the node direction, and then perform whitening on these sub-matrices separately. Thus the feature dimension is balanced.

**Definition 3** *(Node-shuffled whitening). The node-shuffled whitening layer with a group size G takes the l-layer feature matrix* $\mathbf{H}_l \in \mathbb{R}^{N\times d_l}$ *of N nodes as input, which output is another feature matrix* $\mathbf{H}'_l \in \mathbb{R}^{N\times d_l}$ *computed as follows:*

$$\mathbf{H}'_l = \mathrm{NSW}(\mathbf{H}_l) = \mathcal{P}^{-1}\left(\mathrm{ZCA}_G(\mathcal{P}(\mathbf{H}_l))\right) \tag{4}$$

*where* $\mathcal{P}(\cdot)$ *is a random N-order permutation,* $\mathcal{P}(\mathbf{H}_l)$ *is obtained by rearranging rows of* $\mathbf{H}_l$ *according to* $\mathcal{P}$ *and* $\mathcal{P}^{-1}(\mathbf{H}_l)$ *is obtained by rearranging rows according to the inverse permutation of* $\mathcal{P}$.

*In other words, node-shuffled whitening permutes the N node dimensions randomly before applying ZCA whitening with the G node embeddings iteratively and reverses the permutation for outputs.*

To better decorrelate the representation, we apply node-shuffled whitening after each GNN layer in training period. Note that we utilise vanilla ZCA whitening (Definition 2) in inference period to get more informative embeddings.

### 5.3. Whitening-enhancing loss

Dimensional collapse suggests an excessive correlation between embedding dimensions, potentially leading to a limited encoding of useful information. To mitigate this issue, in addition to directly applying whitening to node embeddings, we suggest enriching the encoded information by maximising the mutual information (MI) between input features and node embeddings. This restriction can enhance the effect of whitening.

However, directly maximising the mutual information (MI) between input features and node embeddings is a non-trivial task, as the embeddings are learned through neural networks. To address this issue, we adopt the approach proposed in MINE [40] and DeCorr [41] to estimate a lower bound of the MI by training a classifier to distinguish between sample pairs from the joint distribution $P(A, B)$ and those from $P(A)P(B)$:

$$\mathrm{MI}(A,B) \geq \mathbb{E}_{P(A,B)}[\mathcal{D}(A,B)] - \log \mathbb{E}_{P(A)P(B)}\left[e^{\mathcal{D}(A,B)}\right], \tag{5}$$

where $A, B$ are two random variables, $\mathcal{D}(A, B)$ is a binary discriminator. Considering the valuable information from neighbouring nodes that is similar and homogeneous in nature, we maximise the mutual information between the $l$-th layer embeddings $\mathbf{H}_l$[1] and input feature with $n$-hop neighbour information $\mathbf{X}' = (\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}})^n\mathbf{X}$ ($\widetilde{\mathbf{A}}$ is adjacency matrix with added self-connection, $\widetilde{\mathbf{D}}_{ii} = \sum_j \widetilde{\mathbf{A}}_{ij}$, $n$ is a hyperparameter) based on GCN [1]. We minimise the following objective:

$$\ell_{\mathrm{WE}}\left(\mathbf{H}_l, \mathbf{X}'\right) = -\mathbb{E}_{P(\mathbf{h}_{l,i},\mathbf{x}'_i)}\left[\mathcal{D}\left(\mathbf{h}_{l,i}, \mathbf{x}'_i\right)\right] + \log \mathbb{E}_{P(\mathbf{h}_{l,i})P(\mathbf{x}'_j)}\left[e^{\mathcal{D}\left(\mathbf{h}_{l,i},\mathbf{x}'_j\right)}\right], \tag{6}$$

where $\mathbf{h}_{l,i}$ and $\mathbf{x}_i$ are the $l$-layer node embeddings and input feature with neighbour information for node $v_i$, respectively; the discriminator $\mathcal{D}(\cdot,\cdot)$ is modelled as:

$$\mathcal{D}\left(\mathbf{x}'_i, \mathbf{h}_{l,i}\right) = \sigma\left(\mathbf{x}'^{T}_i \mathbf{W}\mathbf{h}_{l,i}\right), \tag{7}$$

where $\mathbf{W}$ is a learning $d \times d_l$ matrix. In practice, in each epoch, we utilise $\left\{\left(\mathbf{h}_{l,i}, \mathbf{x}_i\right)\right\}_{i=1}^{N}$ from the joint distribution $P\left(\mathbf{h}_{l,i}, \mathbf{x}'_i\right)$ to calculate the first term in Eq. (6) and then shuffle $x_i$ to generate "negative pairs" for **estimating** the second term. Considering that random shuffling can result in a node being paired with itself to form a negative sample, thereby reducing the rationality of the model, we will remove such samples. Then we apply the above loss function for each layer of GNNs and **across views**:

$$\begin{aligned}\mathcal{L}_{\mathrm{WE}} = \sum\nolimits_{l=1}^{L} &\ell_{\mathrm{WE}}\left(\mathbf{H}_l^{(1)}, \mathbf{X}'_1\right) + \ell_{\mathrm{WE}}\left(\mathbf{H}_l^{(2)}, \mathbf{X}'_2\right) + \\ &\ell_{\mathrm{WE}}\left(\mathbf{H}_l^{(1)}, \mathbf{X}'_2\right) + \ell_{\mathrm{WE}}\left(\mathbf{H}_l^{(2)}, \mathbf{X}'_1\right),\end{aligned} \tag{8}$$

where $\mathbf{X}'_* = (\widetilde{\mathbf{D}}_*^{-\frac{1}{2}}\widetilde{\mathbf{A}}_*\widetilde{\mathbf{D}}_*^{-\frac{1}{2}})^n\mathbf{X}_*$ ($*$ is the view 1 or 2) denotes $n$-hop input features of different views.

### 5.4. Overall objective function

By incorporating the whitening-enhancing loss ($\mathcal{L}_{\mathrm{WE}}$) to enrich the encoded information, the overall loss function is derived through the joint optimisation of the InfoNCE loss in Section 3 and $\mathcal{L}_{\mathrm{WE}}$.

$$\mathcal{L} = \mathcal{L}_{\mathrm{NCE}} + \alpha\mathcal{L}_{\mathrm{WE}} \tag{9}$$

where $\alpha$ is the hyper-parameter that controls the contribution of $\mathcal{L}_{\mathrm{WE}}$. The fine-tuning of $\alpha$ proves to be remarkably convenient, and in the subsequent section (Section 6), we shall proceed to its intricate details. The training algorithm is summarised in Algorithm 1.

[1] $\mathbf{H}_l$ here denotes the output of node-shuffled whitening ($\mathbf{H}'_l$ in Definition 3). We adopt the notation $\mathbf{H}_l$ to maintain consistency with the notation used in Section 3.1.

**Algorithm 1:** The baseline+WGCL training algorithm.

**for** $epoch \leftarrow 1, 2, \cdots$ **do**
  // Data augmentation
  Sample two stochastic augmentation functions $t_1 \sim \mathcal{T}$ and $t_2 \sim \mathcal{T}$
  Generate two graph views $\mathcal{G}_1 = (\mathbf{X}_1, \mathbf{A}_1) = t_1(\mathcal{G})$ and $\mathcal{G}_2 = (\mathbf{X}_2, \mathbf{A}_2) = t_2(\mathcal{G})$ by performing corruption on $\mathcal{G}$
  // Obtain node embeddings
  Denote $\mathbf{H}_0^{(1)} = \mathbf{X}_1$, $\mathbf{H}_0^{(2)} = \mathbf{X}_2$
  **for** $view \leftarrow 1, 2$ **do**
    **for** $l \leftarrow 1, 2, \cdots, L$ **do**
      $\mathbf{H}_l^{(view)} = \mathrm{NSW}(f_l(\mathbf{H}_{l-1}^{(view)}, \mathbf{A}_{view}))$
  Denote $\mathbf{H}^{(1)} = \mathbf{H}_L^{(1)}$, $\mathbf{H}^{(2)} = \mathbf{H}_L^{(2)}$
  // Calculate loss function and back propagation
  Compute the InfoNCE loss $\mathcal{L}_{NCE}$ with definition in Section 3.2
  Compute the whitening-enhancing loss $\mathcal{L}_{WE}$ with Eq. (8)
  Compute the overall objective $\mathcal{L}$ with Eq. (9)
  Update parameters by applying stochastic gradient descent to minimise $\mathcal{L}$

**Table 2**
Statistics of the datasets used in the experiments.

| Dataset | Acronym | #Nodes | #Edges | #Features | #Classes |
|---|---|---|---|---|---|
| Wiki-CS [43] | - | 11,701 | 216,123 | 300 | 10 |
| Amazon-Computers [42] | Am-Com | 13,752 | 245,861 | 767 | 10 |
| Amazon-Photo [42] | Am-Pho | 7,650 | 119,081 | 745 | 8 |
| Coauthor-CS [44] | Co-CS | 18,333 | 81,894 | 6,805 | 15 |
| Coauthor-Physics [44] | Co-Phy | 34,493 | 247,962 | 8,415 | 5 |
| Cora [1] | - | 2,708 | 5,429 | 1,433 | 7 |
| Citeseer [1] | - | 3,327 | 4,732 | 3,703 | 6 |
| Pubmed [1] | - | 19,717 | 44,338 | 500 | 3 |
| DBLP [1] | - | 17,716 | 105,734 | 1,639 | 4 |

## 6. Experiments

In alignment with the prevailing trends in GCL research, such as G-BT [14], GCA [13], COSTA [15], and others, our primary focus is on node classification tasks. Below, we present details of our experimental settings and discuss our results. We address the following research questions (RQs):

- **RQ1.** Can WGCL help *improving* the performance of *various* GCL methods in node classification?
- **RQ2.** What role do the two components, i.e., node-shuffled whitening and whitening-enhancing loss, play in decreasing the correlation among the dimensions of node embeddings separately?
- **RQ3.** What is the impact of key hyperparameters on the model's performance?

The code to reproduce the experiments is available at https://github.com/acboyty/WGCL.

### 6.1. Experimental setup

#### 6.1.1. Computing environment

We implement all models using PyTorch Geometric 2.1.0 and PyTorch 1.12.1. The datasets used in our experiments are readily available in PyTorch Geometric libraries. Our experiments are conducted on a computer server with four NVIDIA A40 GPUs, each equipped with 48 GB of memory, and four Intel(R) Xeon(R) Gold 5220 CPUs.

#### 6.1.2. Datasets

To evaluate the effectiveness of our method, we employ nine widely used benchmark datasets from previous research [9,13,16]. These datasets include citation networks such as Cora, CiteSeer, Pubmed and DBLP, as well as social networks such as Wiki-CS, Amazon-Computers, Amazon-Photo, Coauthor-CS and Coauthor-Physics [1,42–44]. We make a description of datasets from Table 2:

- **Cora, CiteSeer, Pubmed, DBLP.** These are widely recognised citation network datasets where each node represents a publication and the edges indicate citation relationships between them. All the nodes in the network are labelled according to the subject area of the corresponding paper [1].
- **Wiki-CS.** It is a network comprised of Wikipedia pages related to computer science, where edges indicate cross-references between articles. Each article is assigned to one of 10 subfields (classes), and the characteristics of the content are computed using the averaged GloVe embeddings [43].

**Table 3**
Results of the node classification on social networks in terms of accuracy (%) with standard deviations. The unsupervised models with the highest accuracy are highlighted in bold, while the supervised models with the highest accuracy are indicated with a wavy underline. The underlined number shows the improvement (↑) or decrease (↓) in accuracy (%) after our WGCL plugin is added to backbones. OOM indicates Out-Of-Memory.

| Method | Wiki-CS | Am-Com | Am-Pho | Co-CS | Co-Phy |
|---|---|---|---|---|---|
| GCN | 77.1±0.1 | 86.5±0.5 | 92.4±0.2 | 93.0±0.3 | 95.6±0.1 |
| GAT | 77.6±0.1 | 86.9±0.2 | 92.5±0.3 | 92.3±0.2 | 95.4±0.1 |
| Raw features | 71.9±0.0 | 73.8±0.0 | 78.5±0.0 | 90.3±0.0 | 93.5±0.0 |
| node2vec | 71.7±0.0 | 84.3±0.0 | 89.6±0.1 | 85.0±0.0 | 91.1±0.0 |
| DeepWalk(DW) | 74.3±0.0 | 85.6±0.0 | 89.4±0.1 | 84.6±0.2 | 91.7±0.1 |
| DW+features | 77.2±0.0 | 86.2±0.0 | 90.0±0.0 | 87.7±0.0 | 94.9±0.0 |
| GAE | 70.1±0.0 | 85.2±0.1 | 91.6±0.1 | 90.0±0.7 | 94.9±0.0 |
| VGAE | 75.6±0.1 | 86.3±0.2 | 92.2±0.1 | 92.1±0.0 | 94.5±0.0 |
| DGI | 75.3±0.1 | 83.9±0.4 | 91.6±0.2 | 92.1±0.6 | 94.5±0.5 |
| GMI | 74.8±0.0 | 82.2±0.3 | 90.6±0.1 | OOM | OOM |
| MVGRL | 77.5±0.0 | 87.5±0.1 | 91.7±0.0 | 92.1±0.1 | 95.3±0.3 |
| MERIT | 78.3±0.0 | 88.0±0.1 | 92.5±0.1 | 92.5±0.1 | OOM |
| G-BT | 76.8±0.7 | 87.9±0.3 | 92.4±0.3 | 92.9±0.2 | 95.2±0.1 |
| GRACE | 78.3±0.0 | 87.8±0.2 | 92.5±0.1 | 92.9±0.0 | 95.7±0.0 |
| GRACE+WGCL | 78.8±0.0↑0.5 | 88.9±0.3↑1.1 | **93.2**±0.0↑0.7 | 93.1±0.0↑0.2 | 95.8±0.0↑0.1 |
| GCA | 78.3±0.0 | 87.8±0.3 | 92.4±0.0 | 93.1±0.0 | 95.6±0.0 |
| GCA+WGCL | 79.5±0.0↑1.2 | **89.5**±0.2↑1.7 | 93.2±0.2↑0.8 | **93.3**±0.1↑0.2 | **95.9**±0.0↑0.3 |
| COSTA | 79.1±0.0 | 88.3±0.0 | 92.5±0.4 | 92.9±0.1 | 95.7±0.2 |
| COSTA+WGCL | **80.3**±0.1↑1.2 | 89.1±0.1↑0.8 | 93.1±0.9↑0.6 | 93.0±0.2↑0.1 | 95.8±0.1↑0.1 |

- **Amazon-Computers, Amazon-Photo.** Both of these networks are constructed based on Amazon's co-purchase data, where nodes represent products and edges indicate how frequently they were purchased together. The products are described using a Bag-of-Words representation based on the reviews, which serve as the node features. In Amazon-Computers and Amazon-Photo datasets, there are ten and eight node classes (product categories), respectively [42].
- **Coauthor-CS, Coauthor-Physics.** These two networks are extracted from the Microsoft Academic Graph dataset, where the edges reflect collaborations between two authors and the nodes represent the authors themselves. The authors are categorised based on the keywords they use in their articles, which are represented using the Bag-of-Words technique as node features. There are 15 author research fields and 5 author research fields, respectively, as the node classes in these two networks [44].

With the exception of Wiki-CS, which adopts a public split, we use a random split setting for remaining datasets, which randomly allocates 10/10/80% of data to training/validation/test set, respectively.

#### 6.1.3. Baselines

We select representative baselines from traditional, autoencoder-based, and contrastive-based graph self-supervised learning. These baselines include: (a) random walk-based models: Deepwalk [45] and node2vec [46]; (b) autoencoder-based models: GAE and VGAE [32]; (c) contrastive-based models: DGI [9], GMI [10], MVGRL [11], MERIT [12], GRACE [13], GCA [16], G-BT [14] and COSTA [15]. In addition, we report the performance of two representative supervised models, GCN [1] and GAT [2]. The performance of all baselines is reported based on either their official implementations or previously published reports.

#### 6.1.4. Evaluation metrics

For each experiment, we employ the same evaluation scheme as in prior studies [9,13,16], where each model is initially trained in an unsupervised manner on the entire graph, utilising node features. Subsequently, we convert the raw features into resulting embeddings using the trained encoder. Finally, we employ an $\ell_2$-regularised logistic regression classifier from the Scikit-Learn library [47], utilising the embeddings obtained in the previous step. We perform five random initialisations of GCL methods and report the mean and standard deviation of the calculated classification accuracy.

### 6.2. Improving various GCL methods (RQ1)

To address **RQ1**, we integrate our proposed WGCL plugin with three leading GCL methods, namely GRACE [13], GCA [16], and COSTA [15] (as backbones). We examine the impact of WGCL on the performance of each backbone and compare the results with those of selected baselines as shown in Table 3 and 4. To ensure reproducibility, we have included detailed information about our hyperparameters in Table 6.

**Comparison with state-of-the-art performances.** Upon analysing the two tables, it can be concluded that our proposed WGCL plugin consistently yields the best results on nine widely-used datasets when incorporated into the backbone models of GRACE, GCA, and COSTA. In terms of performance on the WikiCS, Amazon-Computers, and CiteSeer datasets, bacbones+WGCL demonstrated

**Table 4**
Results on citation networks. We use the same settings as Table 3.

| Method | Cora | CiteSeer | PubMed | DBLP |
|---|---|---|---|---|
| GCN | 81.4±0.5 | 70.9±0.5 | 84.9±0.2 | 83.3±0.2 |
| GAT | 83.0±0.7 | 72.5±0.7 | 84.8±0.3 | 83.9±0.7 |
| Raw features | 64.8±0.0 | 64.6±0.0 | 84.8±0.0 | 71.6±0.0 |
| node2vec | 74.8±0.2 | 52.3±0.5 | 80.3±0.8 | 78.8±0.1 |
| DeepWalk(DW) | 75.7±0.3 | 50.5±0.4 | 80.5±0.2 | 75.9±0.5 |
| DW+features | 73.1±0.8 | 47.6±0.1 | 83.7±0.3 | 78.1±0.4 |
| GAE | 76.9±0.3 | 60.6±0.4 | 82.9±0.7 | 81.2±0.6 |
| VGAE | 78.9±0.5 | 61.2±0.2 | 83.0±0.3 | 81.7±0.4 |
| DGI | 82.6±0.4 | 68.8±0.7 | 86.0±0.1 | 83.2±0.1 |
| GMI | 83.0±0.3 | 72.4±0.1 | 79.9±0.2 | 82.7±0.3 |
| MVGRL | 84.3±0.2 | 73.3±0.5 | 80.1±0.7 | 83.1±0.3 |
| MERIT | 83.1±0.6 | 74.0±0.7 | 80.1±0.4 | 83.9±0.2 |
| G-BT | 83.9±0.3 | 72.7±0.3 | 86.4±0.2 | 84.8±0.1 |
| GRACE | 83.3±0.4 | 72.1±0.5 | 86.6±0.1 | 84.2±0.2 |
| GRACE+WGCL | 84.3±0.5↑1.0 | 72.9±0.4↑0.8 | 86.7±0.1↑0.1 | 84.5±0.1↑0.3 |
| GCA | 83.6±0.2 | 72.4±0.3 | 86.5±0.2 | 83.9±0.2 |
| GCA+WGCL | 84.4±0.4↑0.7 | 73.0±0.3↑0.6 | **87.0**±0.1↑0.5 | **84.7**±0.3↑0.8 |
| COSTA | 83.9±0.1 | 72.8±0.3 | 86.3±0.1 | 84.5±0.1 |
| COSTA+WGCL | **84.7**±0.8↑0.8 | **74.8**±0.1↑2.0 | 86.4±0.4↑0.1 | 84.6±0.2↑0.1 |

improvements of 1.3%, 1.3%, and 2.0%, respectively, compared to the unsupervised baselines. When comparing GCA+WGCL with all unsupervised baselines, it achieved an average improvement of 0.5%. This remarkable performance confirms the effectiveness of our WGCL plugin. Notably, even though the existing methods had high accuracy on the Coauthor datasets, we observed that the backbones+WGCL still outperforms them. Furthermore, we observe that the performance of backbones+WGCL was on par with models trained with label supervision on all five datasets, which demonstrates its exceptional performance.

**The effectiveness of WGCL plugin.** To evaluate the impact of our plugin, WGCL, on the performance of three backbone models, we conducted experiments to measure the percentage increase (↑) in accuracy or decrease (↓) in percentage points. Our results consistently demonstrate that integrating WGCL significantly improves the performance of all three backbone models, showcasing its versatility as a plug-and-play module. Specifically, on the Wiki-CS, Amazon-Computers, and CiteSeer datasets, our approach, WGCL achieved an average improvement of 1.0%, 1.2%, and 1.1% across all backbones. Moreover, when evaluated on the challenging Coauthor-CS and Coauthor-Physics datasets, WGCL achieved an average improvement of 0.2% and 0.07%, respectively, across all backbones. We attribute this improvement to the whitening approach employed in our plugin, which effectively addresses the issue of dimensional collapse of node embeddings. By mitigating the loss of information due to the collapse of dimensions, our approach enables better differentiation of each embedding and consequently, superior performance in downstream classification tasks. Our experimental results strongly suggest that the proposed plugin can be seamlessly integrated with various GCL methods, enabling significant performance improvements.

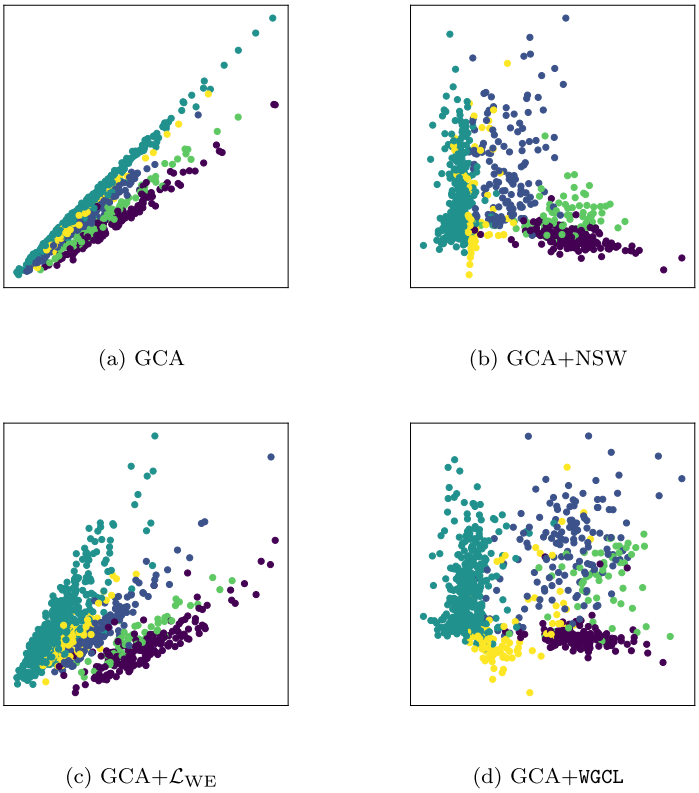
### 6.3. Ablation study (RQ2)

**NSW and $\mathcal{L}_{\mathrm{WE}}$ effectively eliminates dimensional collapse.** We visualise the 2D embedding space in Fig. 4 where the training distribution of Amazon-Computers are represented. To demonstrate the efficacy of whitening, we consider four settings of experiments: (1) using only the InfoNCE loss, (2) incorporating the node-shuffled whitening (NSW), (3) incorporating the whitening-enhancing loss ($\mathcal{L}_{\mathrm{WE}}$), and (4) incorporating both improvements (WGCL). We could observe that training with only InfoNCE loss suffers greatly from dimensional collapse, which results in node embeddings of different categories hard to distinguish. As we incorporate NSW and $\mathcal{L}_{\mathrm{WE}}$ separately, dimensional collapse is alleviated to varying degrees. Finally, we find that utilising NSW and $\mathcal{L}_{\mathrm{WE}}$ simultaneously could fully solve the dimensional collapse issue, which demonstrates the effectiveness of our method.

**Performance with/without sub-modules.** To further demonstrate the effectiveness of whitening, we conduct experiments removing various parts of WGCL to study the impact of each component shown in Table 5. Note that instead of 2D latent space, we utilise the same settings as Table 3. Considering that whitening includes normalisation, we consider batch normalisation as a separate variable, from which we can see the unique role of whitening. We can see that normalisation cannot stably improve the performance of the baseline, mainly because it cannot solve the problem of dimensional collapse. NSW and $\mathcal{L}_{\mathrm{WE}}$ generally have a promotion effect on the backbone, and the combination of the two has reached the maximum.

### 6.4. Hyperparameters analysis (RQ3)

**Hyperparameter settings.** All models use Glorot initialisation for parameter initialisation and are optimised using the Adam optimiser with weight decay factor $\ell_2 = 10^{-5}$ and dropout rate of 0 across all datasets. The probability parameters for controlling the sampling process, including $p_{e,1}, p_{f,1}$ for the first view and $p_{e,2}, p_{f,2}$ for the second view, $p_\tau, \tau$ and activation function, are kept



(a) GCA (b) GCA+NSW

(c) GCA+$\mathcal{L}_{\mathrm{WE}}$ (d) GCA+WGCL

**Fig. 4.** The training distribution in our 2D embedding space. Dataset: Amazon-Computers.

**Table 5**
Ablation study of different component.

| Method | Wiki-CS | Am-Com | Am-Pho | Co-CS | Co-Phy |
|---|---|---|---|---|---|
| GCA | 78.30 | 87.85 | 92.49 | 93.10 | 95.68 |
| +BN | 78.12↓0.18 | 88.44↑0.59 | 93.18↑0.69 | 92.79↓0.31 | 95.45↓0.23 |
| +NSW | 79.39↑1.09 | 89.19↑1.34 | 93.20↑0.71 | 93.05↓0.05 | 95.79↑0.11 |
| + $\mathcal{L}_{\mathrm{WE}}$ | 79.53↑1.23 | 89.10↑1.25 | 92.89↑0.40 | 93.22↑0.12 | 95.69↑0.01 |
| +WGCL | **79.54**↑1.24 | **89.57**↑1.72 | **93.24**↑0.75 | **93.37**↑0.27 | **95.82**↑0.14 |
| COSTA | 79.12 | 88.32 | 92.56 | 92.95 | 95.74 |
| +BN | 79.22↑0.10 | 88.89↑0.57 | 92.45↓0.11 | 92.79↓0.31 | 95.45↓0.23 |
| +NSW | 79.86↑0.74 | 89.11↑0.79 | 92.98↑0.42 | 92.97↑0.02 | 95.68↓0.06 |
| + $\mathcal{L}_{\mathrm{WE}}$ | 79.23↑0.11 | 89.03↑0.71 | 92.79↑0.23 | 93.08↑0.13 | 95.75↑0.01 |
| +WGCL | **80.39**↑1.27 | **89.12**↑0.80 | **93.12**↑0.56 | **93.05**↑0.10 | **95.76**↑0.02 |

**Table 6**
Hyperparameter specifications.

| Dataset | $p_{e,1}$ | $p_{e,2}$ | $p_{f,1}$ | $p_{f,2}$ | $p_\tau$ | $\tau$ | Activation function | Learning rate | Training epochs | Hidden dimension | $n$ | $\alpha$ | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cora | 0.3 | 0.4 | 0.2 | 0.4 | - | 0.7 | ReLU | 5e-4 | 50 | 512 | 2 | 10/0.1/0.1 | 5 |
| Citeseer | 0.3 | 0.2 | 0.2 | 0.0 | - | 0.7 | PReLU | 5e-4 | 30 | 2,048 | 2 | 1/5/10 | 5 |
| Pubmed | 0.0 | 0.2 | 0.4 | 0.1 | - | 0.7 | ReLU | 5e-4 | 200 | 512 | 2 | 0.01/0.01/0.01 | 5 |
| DBLP | 0.1 | 0.0 | 0.1 | 0.4 | - | 0.7 | ReLU | 5e-4 | 1,000 | 600 | 2 | 10/10/5 | 5 |
| Wiki-CS | 0.2 | 0.4 | 0.1 | 0.1 | 0.7 | 0.6 | PReLU | 1e-3 | 300 | 256 | 2 | 5/0.1/0.1 | 5 |
| Amazon-Computers | 0.5 | 0.5 | 0.2 | 0.1 | 0.7 | 0.1 | PReLU | 1e-3 | 400 | 128 | 2 | 100/0.1/0.01 | 5 |
| Amazon-Photo | 0.3 | 0.5 | 0.1 | 0.1 | 0.7 | 0.3 | ReLU | 1e-3 | 400 | 256 | 2 | 10/5/0.1 | 5 |
| Coauthor-CS | 0.3 | 0.2 | 0.3 | 0.4 | 0.7 | 0.4 | RReLU | 1e-4 | 300 | 256 | 2 | 10/100/10 | 5 |
| Coauthor-Physics | 0.4 | 0.1 | 0.1 | 0.4 | 0.7 | 0.5 | RReLU | 5e-4 | 100 | 512 | 2 | 1/0.1/5 | 5 |

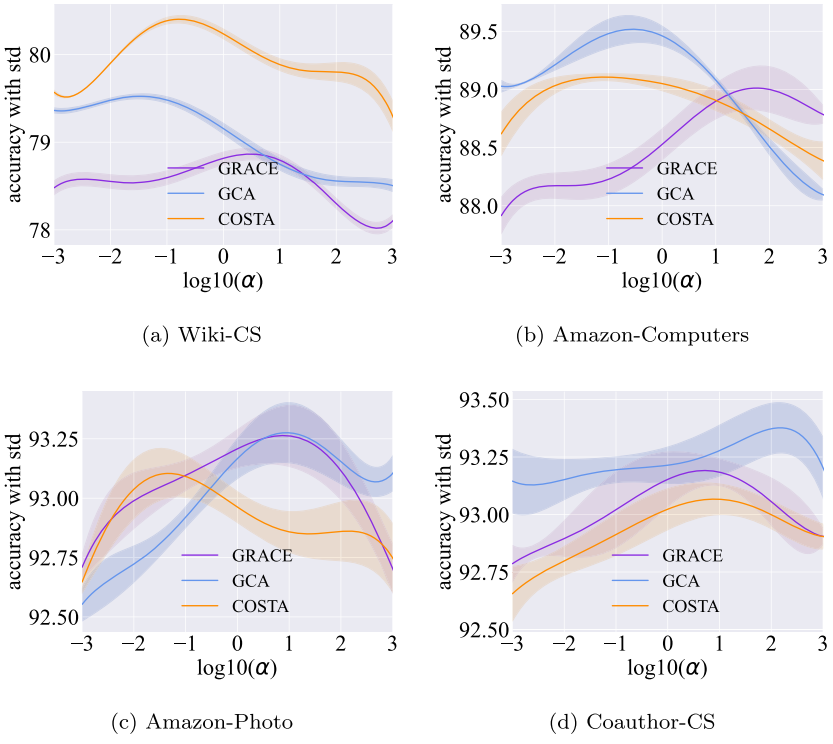



**Fig. 5.** The performance of backbones (GRACE/GCA/COSTA)+WGCL with regard to changes of $\alpha$.

consistent with GRACE [13] and GCA [16]. We varied the number of training epochs, learning rate, and hidden dimensions for the three backbone architectures. The hop $n$ and $\alpha$ is also provided (GRACE/GCA/COSTA). The number $T$ of Newton's iteration required to estimate the precise whitening matrix is also given. A summary of the dataset-specific hyperparameter configurations can be found in Table 6. From the table, WGCL greatly speeds up the convergence speed of the model, compared with training epochs provided in [13,16,15].

**Sensitive analysis on $\alpha$.** We performed a comprehensive sensitivity analysis on the crucial hyperparameter $\alpha$ of WGCL by varying its values across several orders of magnitude, ranging from $10^{-3}$ to $10^{3}$. We fit the curves in Fig. 5, where the horizontal axis is logarithmically scaled to base 10. During the sensitivity analysis, we kept all other parameters constant as previously described, and only adjusted the value of $\alpha$. From Fig. 5, we observe that fine-tuning the value of $\alpha$ is a relatively straightforward task. 1) The accuracy generally increases first and then decreases as the value of $\alpha$ increases on different datasets, thus a one-dimensional grid search suffices for finding the optimal $\alpha$. 2) Most optimal alpha values fall within $10^{-2}$ to $10^{2}$, narrowing the search space significantly. 3) WGCL is robust to $\alpha$. Consider the ordinate scale. Despite a wide range of $\alpha$ values from $10^{-3}$ to $10^{3}$, corresponding performance changes are minor. Simply setting $\alpha$ to 1 produces satisfactory results. With meticulous consideration, we utilise grid search to identify the optimal value of $\alpha$ for different datasets, as indicated in Table 6.

### 6.5. Discussion and future work

We below present a discussion on the sensitivity to dataset properties, in which the future work also is analysed.

**Sensitivity to dataset properties.** The degree of improvement achieved by WGCL varies across different datasets. We analyse the reasons behind this phenomenon. 1) The improvement space of different data sets varies. This leads to variations in WGCL's performance across datasets. 2) WGCL aims to eliminate correlations between feature dimensions. If the original method already exhibits weak correlations on a few datasets, WGCL may result in marginal improvements. However, on the majority of datasets, WGCL consistently achieves substantial enhancements.

**Broader Scope of Applications.** The potential applications of WGCL have not been fully explored. Our plugin exhibits adaptability to a wide range of datasets from diverse domains. In this paper, we have incorporated datasets from social networks and citation networks, which are already highly representative. Thus there is one promising avenue for future work: applying GCL with WGCL to more real-world tasks including social analysis, cyber-security, federated learning and biochemistry. Furthermore, given the effectiveness of WGCL across various GCL methods, whether WGCL or its enhanced version can prove effective in the context of graph weak-supervised learning or semi-supervised learning is a topic worthy of investigation.

## 7. Conclusion

In graph contrastive learning (GCL), the issue of dimensional collapse occurs when the embeddings are concentrated on only a few dimensions, which can be inevitably caused by the InfoNCE loss used in typical GCL. To solve this problem, Whitening GCL (WGCL) has been proposed, which uses node-shuffled whitening to decorrelate the dimensions of the embeddings and distribute them evenly in the embedding space. Additionally, a whitening-enhancing loss function is employed to prevent a decrease in information due to dimensional collapse and to enhance the model's representational capacity. Experiment results demonstrate that WGCL effectively relieves dimensional collapse, leading to significant improvements across various GCL backbones on widely-used datasets.

## CRediT authorship contribution statement

**Yang Tao:** Conceptualization, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Kai Guo:** Conceptualization, Writing – review & editing. **Yizhen Zheng:** Conceptualization, Writing – review & editing. **Shirui Pan:** Supervision, Writing – review & editing. **Xiaofeng Cao:** Supervision, Writing – review & editing. **Yi Chang:** Data curation, Supervision, Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Acknowledgements

This work was supported in part by National Natural Science Foundation of China, Grant Number: 61976102, U19A2065, 62206108 and Jilin University organised scientific research project, Grant Number: 45123031J0.

## References

[1] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907, 2016.
[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903, 2017.
[3] P. Bachman, R.D. Hjelm, W. Buchwalter, Learning representations by maximizing mutual information across views, Adv. Neural Inf. Process. Syst. 32 (2019).
[4] K. He, H. Fan, Y. Wu, S. Xie, R. Girshick, Momentum contrast for unsupervised visual representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9729–9738.
[5] Y. Tian, D. Krishnan, P. Isola, Contrastive multiview coding, in: European Conference on Computer Vision, Springer, 2020, pp. 776–794.
[6] R. Collobert, J. Weston, A unified architecture for natural language processing: deep neural networks with multitask learning, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 160–167.
[7] A. Mnih, K. Kavukcuoglu, Learning word embeddings efficiently with noise-contrastive estimation, Adv. Neural Inf. Process. Syst. 26 (2013).
[8] T. Hua, W. Wang, Z. Xue, S. Ren, Y. Wang, H. Zhao, On feature decorrelation in self-supervised learning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 9598–9608.
[9] P. Velickovic, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax, in: ICLR, Poster 2 (2019) 4.
[10] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, J. Huang, Graph representation learning via graphical mutual information maximization, in: Proceedings of the Web Conference 2020, 2020, pp. 259–270.
[11] K. Hassani, A.H. Khasahmadi, Contrastive multi-view representation learning on graphs, in: International Conference on Machine Learning, PMLR, 2020, pp. 4116–4126.
[12] M. Jin, Y. Zheng, Y.-F. Li, C. Gong, C. Zhou, S. Pan, Multi-scale contrastive Siamese networks for self-supervised graph representation learning, arXiv preprint arXiv:2105.05682, 2021.
[13] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Deep graph contrastive representation learning, arXiv preprint arXiv:2006.04131, 2020.
[14] P. Bielak, T. Kajdanowicz, N.V. Chawla, Graph Barlow twins: a self-supervised representation learning framework for graphs, Knowl.-Based Syst. 256 (2022) 109631.
[15] Y. Zhang, H. Zhu, Z. Song, P. Koniusz, I. King Costa, Covariance-preserving feature augmentation for graph contrastive learning, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 2524–2534.
[16] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Graph contrastive learning with adaptive augmentation, in: Proceedings of the Web Conference 2021, 2021, pp. 2069–2080.
[17] L. Jing, P. Vincent, Y. LeCun, Y. Tian, Understanding dimensional collapse in contrastive self-supervised learning, arXiv preprint arXiv:2110.09348, 2021.
[18] J. Sun, J. Yan, C. Wu, Y. Ding, R. Chen, X. Yu, X. Lu, J. Li, Graphcoco: graph complementary contrastive learning, arXiv preprint arXiv:2203.12821, 2022.
[19] L. Huang, D. Yang, B. Lang, J. Deng, Decorrelated batch normalization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 791–800.
[20] P. Jia, J. Kou, J. Liu, J. Dai, H. Luo Srfa-grl, Predicting group influence in social networks with graph representation learning, Inf. Sci. 638 (2023) 118960.
[21] F. Wang, Z. Zheng, Y. Zhang, Y. Li, K. Yang, C. Zhu, To see further: knowledge graph-aware deep graph convolutional network for recommender systems, Inf. Sci. 647 (2023) 119465.
[22] H.-C. Yi, Z.-H. You, D.-S. Huang, C.K. Kwoh, Graph representation learning in bioinformatics: trends, methods and applications, Brief. Bioinform. 23 (2022) bbab340.
[23] M.M. Akhter, S.K. Mohanty, A fast o (nlgn) time hybrid clustering algorithm using the circumference proximity based merging technique for diversified datasets, Eng. Appl. Artif. Intell. 125 (2023) 106737.

[24] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: International Conference on Machine Learning, PMLR, 2020, pp. 1597–1607.
[25] Z. Yang, Y. Cheng, Y. Liu, M. Sun, Reducing word omission errors in neural machine translation: a contrastive learning approach, 2019.
[26] H. Li, Y. Gong, J. Jiao, R. Zhang, T. Baldwin, N. Duan, Kfcnet: knowledge filtering and contrastive learning for generative commonsense reasoning, in: Findings of the Association for Computational Linguistics: EMNLP 2021, 2021, pp. 2918–2928.
[27] G. Chu, X. Wang, C. Shi, X. Jiang, Cuco: graph representation with curriculum contrastive learning, in: IJCAI, 2021, pp. 2300–2306.
[28] S. Suresh, P. Li, C. Hao, J. Neville, Adversarial graph augmentation to improve graph contrastive learning, Adv. Neural Inf. Process. Syst. 34 (2021) 15920–15933.
[29] Y. You, T. Chen, Y. Shen, Z. Wang, Graph contrastive learning automated, in: International Conference on Machine Learning, PMLR, 2021, pp. 12121–12132.
[30] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al., Bootstrap your own latent-a new approach to self-supervised learning, Adv. Neural Inf. Process. Syst. 33 (2020) 21271–21284.
[31] X. Chen, K. He, Exploring simple Siamese representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 15750–15758.
[32] T.N. Kipf, M. Welling, Variational graph auto-encoders, arXiv preprint arXiv:1611.07308, 2016.
[33] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, Adv. Neural Inf. Process. Syst. 31 (2018).
[34] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph cnn for learning on point clouds, ACM Trans. Graph. 38 (2019) 1–12.
[35] A.v.d. Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, arXiv preprint arXiv:1807.03748, 2018.
[36] H. Wei, R. Xie, H. Cheng, L. Feng, B. An, Y. Li, Mitigating neural network overconfidence with logit normalization, in: International Conference on Machine Learning, PMLR, 2022, pp. 23631–23644.
[37] L. Huang, X. Liu, B. Lang, A. Yu, Y. Wang, B. Li, Orthogonal weight normalization: solution to optimization over multiple dependent Stiefel manifolds in deep neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018.
[38] D.A. Bini, N.J. Higham, B. Meini, Algorithms for the matrix p th root, Numer. Algorithms 39 (2005) 349–378.
[39] L. Huang, Y. Zhou, F. Zhu, L. Liu, L. Shao, Iterative normalization: beyond standardization towards efficient whitening, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4874–4883.
[40] M.I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, R.D. Hjelm, Mine: mutual information neural estimation, arXiv preprint arXiv:1801.04062, 2018.
[41] W. Jin, X. Liu, Y. Ma, C. Aggarwal, J. Tang, Feature overcorrelation in deep graph neural networks: a new perspective, arXiv preprint arXiv:2206.07743, 2022.
[42] J. McAuley, C. Targett, Q. Shi, A. Van Den Hengel, Image-based recommendations on styles and substitutes, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2015, pp. 43–52.
[43] P. Mernyei, C. Cangea, Wiki-cs: a Wikipedia-based benchmark for graph neural networks, arXiv preprint arXiv:2007.02901, 2020.
[44] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. Hsu, K. Wang, An overview of microsoft academic service (mas) and applications, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 243–246.
[45] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.
[46] A. Grover, J. Leskovec, node2vec: scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864.
[47] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.