Negative-Free Self-Supervised Gaussian Embedding of Graphs

Yunhui Liu[a], Tieke He[a,*], Tao Zheng[a], Jianhua Zhao[a]

[a]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023, China

Abstract

Graph Contrastive Learning (GCL) has recently emerged as a promising graph self-supervised learning framework for learning discriminative node representations without labels. The widely adopted objective function of GCL benefits from two key properties: *alignment* and *uniformity*, which align representations of positive node pairs while uniformly distributing all representations on the hypersphere. The uniformity property plays a critical role in preventing representation collapse and is achieved by pushing apart augmented views of different nodes (negative pairs). As such, existing GCL methods inherently rely on increasing the quantity and quality of negative samples, resulting in heavy computational demands, memory overhead, and potential class collision issues. In this study, we propose a negative-free objective to achieve uniformity, inspired by the fact that points distributed according to a normalized isotropic Gaussian are uniformly spread across the unit hypersphere. Therefore, we can minimize the distance between the distribution of learned representations and the isotropic Gaussian distribution to promote the uniformity of node representations. Our method also distinguishes itself from other approaches by eliminating the need for a parameterized mutual information estimator, an additional projector, asymmetric structures, and, crucially, negative samples. Extensive experiments over seven graph benchmarks demonstrate that our proposal achieves competitive performance with fewer parameters, shorter training times, and lower memory consumption compared to existing GCL methods.

Keywords: Graph Neural Networks, Graph Representation Learning, Self-Supervised Learning, Graph Data Mining

1. Introduction

One of the current main bottlenecks in graph machine learning is the dependence on heavy annotated training data. Learning representations on graphs without manual labels offers considerable advantages, as highlighted in recent literature [1, 2]. Against this backdrop, Graph Self-Supervised Learning (GSSL) has emerged as a pivotal methodology, addressing this critical need. GSSL demonstrates capabilities that are on par with, or in some cases, surpass those of supervised learning methods, as evidenced by studies such as [3, 4, 5, 6, 7, 8, 9, 10]. Central to the GSSL approach is pre-training, which utilizes meticulously designed pretext objectives. These objectives are task-agnostic, enabling the optimization process to yield representations that are not only general and meaningful but also transferable across various downstream applications.

As a prominent branch of the GSSL family, graph contrastive learning (GCL) [4, 11, 12, 13, 14, 15] has demonstrated remarkable performance and garnered widespread interest. GCL methods focus on learning node representations by creating two augmented views of the input graph and maximizing the mutual information between the encoded representations. Wang and Isola [16] offer an intuitive and theoretical understanding of contrastive learning, emphasizing *alignment* and *uniformity* on the hypersphere. *Alignment*, i.e., pulling together positive pairs, ensures that samples forming positive pairs are mapped to nearby representations, thus rendering them invariant to irrelevant noise factors. However, relying solely on alignment could lead to complete collapse, where all representations converge to a single point [17]. In such cases, the learned representations may exhibit optimal alignment but fail to provide meaningful information for downstream tasks. *Uniformity* assesses how representations are uniformly distributed across the hypersphere, with higher uniformity indicating that more information is preserved in the learned representations. Therefore, *uniformity* plays a pivotal role in alleviating complete collapse and generating discriminative representations [16].

However, existing GCL methods achieve uniformity by pushing apart augmented views of different nodes (negative pairs). As such, they inherently rely on both the quantity and quality of negative samples. This reliance results in substantial computational and memory overhead, as well as class collision, where different samples from the same class are inaccurately deemed negative pairs, thereby impeding representation learning [18, 19]. To address these issues, recent self-supervised methods [20, 19, 21, 22] have explored the prospect of learning without negative samples. Specifically, CCA-SSG [20] and G-BT [21] learn augmentation-invariant information while introducing feature decorrelation to capture orthogonal features and prevent model collapse. However, they may not work well on datasets with low feature dimensions, as they essentially perform dimension reduction. BGRL [19] and AFGRL [22] introduce an online network along with a target network to avoid collapse. But they require additional components, e.g., an ex-

*Corresponding author
Email addresses:* `lyhcloudy1225@gmail.com` (Yunhui Liu), `hetieke@gmail.com` (Tieke He)

ponential moving average (EMA) and Stop-Gradient, leading to a more intricate architecture.

Different from prior works on contrastive learning, we propose Negative-Free Self-Supervised Gaussian Embedding of Graphs (SSGE), a simple yet effective approach that introduces a new negative-free self-supervised learning objective while liberating the model from intricate designs. Following established practices, SSGE generates two views of an input graph through random augmentation and obtains node representations via a shared Graph Neural Network (GNN) encoder. Moreover, our contribution lies in proposing a negative-free self-supervised learning objective. Specifically, this new objective seeks to maximize the agreement between two augmented views of the same input (*alignment*) while simultaneously minimizing the distance between the distribution of learned representations and the isotropic Gaussian distribution (*uniformity*). Our motivation is grounded in the fact that the normalized isotropic Gaussian distributed points are uniformly distributed on the unit hypersphere. The proposed objective yields a simple and light model without depending on negative pairs [4, 11, 12], a parameterized mutual information estimator [3, 23], an additional projector [4, 11, 19, 22], or asymmetric architectures [19, 22, 24, 6]. Our model also works better on low-dimensional datasets than other simple models [20, 21]. Extensive experiments on node classification and node clustering demonstrate that our model consistently achieves highly competitive performance. Furthermore, our method exhibits advantages such as fewer parameters, shorter training times, and lower memory consumption when compared to existing GCL methods. The implementation code is available at `https://github.com/Cloudy1225/SSGE`. To sum up, our contributions are as follows:

- A negative-free uniformity objective is proposed, which is inspired by the fact that points distributed according to a normalized isotropic Gaussian are uniformly spread across the unit hypersphere.

- The proposed objective liberates the self-supervised learning model from the reliance on negative samples and intricate components, including a parameterized mutual information estimator, an additional projector, or asymmetric architectures.

- Extensive experiments across seven graph datasets and two downstream tasks demonstrate that our model achieves competitive performance with fewer parameters, shorter training times, and lower memory consumption than existing GCL methods.

2. Related Works

Recently, numerous research efforts have been devoted to multi-view graph self-supervised learning, which optimizes model parameters by ensuring consensus among multiple views derived from the same sample under different graph augmentations [1]. A crucial aspect of these methods is the prevention of trivial solutions, where all representations converge either to a constant point (i.e., complete collapse) or to a subspace (i.e.,

dimensional collapse). The existing methods can be broadly classified into two groups: contrastive and non-contrastive approaches, each delineated by its strategy for mitigating model collapse.

Contrastive-based methods typically follow the criterion of mutual information maximization [25], whose objective functions involve contrasting positive pairs with negative ones. Pioneering works, such as DGI [3] and GMI [26], focus on unsupervised representation learning by maximizing mutual information between node-level representations and a graph summary vector. MVGRL [23] proposes to learn both node-level and graph-level representations by performing node diffusion and contrasting node representations to augmented graph representation. GRACE [4] and its variants like GCA [11], gCooL [12], and COSTA [13] learn node representations by pulling together the representations of the same node (positive pairs) in two augmented views while pushing away the representations of the other nodes (negative pairs) in two views. AUAR [27] aligns the representations of the node with itself and its cluster centroid while maximizing the distance between nodes and each cluster centroid. DirectCLR [28] directly optimizes the representation space without relying on a trainable projector to mitigate the dimensional collapse in contrastive learning.

Non-contrastive methods eliminate the use of negative samples and adopt different strategies to avoid collapsed solutions. Distillation-based methods BGRL [19], AFGRL [22] introduce an online network along with a target network, where the target network is updated with a moving average of the online network to avoid collapse. GraphALU [24] further captures the uniformity by maximizing the distance between any nodes and a virtual center node. RGRL [29] learns augmentation-invariant relationship, which allows the node representations to vary as long as the relationship among the nodes is preserved. Feature decorrelation methods CCA-SSG [20] and G-BT [21] rely on regularizing the empirical covariance matrix of the representations to capture orthogonal features and prevent dimensional collapse. W-MSE [30] whitens and projects embeddings to the unit sphere before maximizing cosine similarity between positive samples.

Although these methods have demonstrated impressive performance, their reliance on intricate designs and architectures is noteworthy. For example, DGI [3], GMI [26] and MVGRL [23] rely on a parameterized Jensen-Shannon mutual information estimator [31] for distinguishing positive node-graph pairs from negative ones. GRACE [4], GCA [11], gCooL [12], and COSTA [13] harness an additional MLP-projector to guarantee sufficient capacity. Moreover, they necessitate a substantial number of negative samples to prevent model collapse and learn discriminative representations, making them suffer seriously from heavy computation, memory overhead, and class collision [18]. BGRL [19], AFGRL [22], and GraphALU [24] require asymmetric encoders, an exponential moving average and Stop-Gradient, to empirically avoid degenerated solutions, resulting in a more complex architecture. CCA-SSG [20] and G-BT [21] may exhibit suboptimal performance on datasets where input data does not have a large feature dimension, as they are essentially performing dimension reduction. In con-

trast, our method aims to make the distribution of learned representations close to the isotropic Gaussian distribution to achieve uniformity while aligning the representations of two views from data augmentation.

3. Preliminary

3.1. Problem Statement

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represent an attributed graph, where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denote the node set and the edge set, respectively. The graph $\mathcal{G}$ is associated with a feature matrix $X \in \mathbb{R}^{n \times p}$, where $x_i \in \mathbb{R}^p$ represents the feature of v_i, and an adjacency matrix $A \in \{0, 1\}^{n \times n}$, where $A_{i,j} = 1$ if and only if $(v_i, v_j) \in \mathcal{E}$. In the self-supervised training setting, no task-specific labels are provided for $\mathcal{G}$. The primary objective is to learn an embedding function $f_\theta(A, X)$ that transforms X to Z, where $Z \in \mathbb{R}^{n \times d}$ and $d \ll p$. The pre-trained representations aim to capture both attribute and structural information inherent in $\mathcal{G}$ and are easily transferable to various downstream tasks, such as node classification and node clustering.

3.2. Graph Convolutional Network

The Graph Convolutional Network (GCN) [32] is one of the most popular graph neural networks. It is a layer-wise propagation rule-based model to learn the node representation $z_i \in \mathbb{R}^d$ corresponding to node v_i. The formulation of the graph convolutional layer can be expressed as:

$$Z^{(l+1)} = \sigma\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}Z^{(l)}\Theta^{(l)}\right), \qquad (1)$$

where $Z^{(l+1)}$ denotes the node representations at the $l + 1$ layer and $Z^{(0)}$ represents the original attribute matrix of nodes. $\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ is a symmetric normalization of A with self-loop, $\hat{A} = A + I$. I and $\hat{D}$ are the identity matrix and the diagonal node degree matrix of $\hat{A}$, respectively. Additionally, $\Theta^{(l)}$ represents the weight matrix at the l-th layer, and σ denotes the activation function. Consistent with previous works [3, 4, 20, 12, 5], we adopt GCN as the foundational graph encoder.

3.3. Wasserstein Distance

Wasserstein distances are metrics quantifying the dissimilarity between probability distributions, drawing inspiration from the optimal transportation problem [33]. The p-Wasserstein distance is formulated as follows:

$$\mathcal{W}_p(\mathbb{P}_r, \mathbb{P}_g) = \left(\inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma}\left[\|x - y\|^p\right]\right)^{\frac{1}{p}}, \qquad (2)$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ is the set of all joint distributions $\gamma(x, y)$ whose marginals are $\mathbb{P}_r$ and $\mathbb{P}_g$, respectively. The term $\gamma(x, y)$ intuitively indicates the amount of "mass" requiring transportation from x to y for transforming the distribution $\mathbb{P}_r$ into $\mathbb{P}_g$. The Wasserstein distance thus represents the "cost" associated with the optimal transport plan.

In the case of considering both distributions as multivariate Gaussian distributions, i.e., $\mathbb{P}_r = \mathcal{N}(\mu_1, \Sigma_1)$ and $\mathbb{P}_g = \mathcal{N}(\mu_2, \Sigma_2)$, with mean vectors μ_1, μ_2, and covariance matrices Σ_1, Σ_2, respectively, the 2-Wasserstein distance has a closed form expression given by

$$\mathcal{W}_2^2(\mathbb{P}_r, \mathbb{P}_g) = \|\mu_1 - \mu_2\|_2^2 + \text{Tr}\left(\Sigma_1 + \Sigma_2 - 2\left(\Sigma_2^{\frac{1}{2}}\Sigma_1\Sigma_2^{\frac{1}{2}}\right)^{\frac{1}{2}}\right), \quad (3)$$

where $\text{Tr}(\cdot)$ denotes the trace of a matrix. This equation illustrates that the 2-Wasserstein distance between two Gaussian distributions can be easily computed.

4. Methodology

4.1. Model Framework

Our model is simply constructed with three key components: 1) a random graph augmentation generator $\mathcal{T}$, 2) a GNN-based graph encoder symbolized as f_θ, where θ representing its parameters, and 3) a Gaussian distribution guided objective function. Figure 1 is an illustration of the proposed model.

4.1.1. Graph Augmentation

The augmentation of graph data is a critical component of graph contrastive learning, as it generates diverse graph views, resulting in more generalized representations that are robust against variance. In this study, we jointly adopt two widely utilized strategies, feature masking and edge dropping, to enhance graph attributes and topology information, respectively [4, 11, 20, 19, 12].

Feature Masking. We randomly select a portion of the node features' dimensions and mask their elements with zeros. Formally, we first sample a random vector $\widetilde{m} \in \{0, 1\}^F$, where each dimension is drawn from a Bernoulli distribution with probability $1 - p_m$, i.e., $\widetilde{m}_i \sim \mathcal{B}(1 - p_m), \forall i$. Then, the masked node features $\widetilde{X}$ are computed by $\|_{i=1}^N x_i \odot \widetilde{m}$, where $\odot$ denotes the Hadamard product and $\|$ represents the stack operation (i.e., concatenating a sequence of vectors along a new dimension).

Edge Dropping. In addition to feature masking, we randomly drop a certain fraction of edges from the original graph. Formally, since we only remove existing edges, we first sample a random masking matrix $\widetilde{M} \in \{0, 1\}^{N \times N}$, with entries drawn from a Bernoulli distribution $\widetilde{M}_{i,j} \sim \mathcal{B}(1 - p_d)$ if $A_{i,j} = 1$ for the original graph, and $\widetilde{M}_{i,j} = 0$ otherwise. Here, p_d represents the probability of each edge being dropped. The corrupted adjacency matrix can then be computed as $\widetilde{A} = A \odot \widetilde{M}$.

4.1.2. Training and Inference

During each training epoch, we first select two random augmentation functions, $t^1 \sim \mathcal{T}$ and $t^2 \sim \mathcal{T}$, where $\mathcal{T}$ is composed of all the possible graph transformation operations. Subsequently, two different views, $(\widetilde{A^1}, \widetilde{X^1}) = t^1(A, X)$ and $(\widetilde{A^2}, \widetilde{X^2}) = t^2(A, X)$, are generated based on the sampled functions. These two augmented views are then fed into a shared encoder f_θ to extract the corresponding node representations: $Z^1 = f_\theta(\widetilde{A^1}, \widetilde{X^1})$ and $Z^2 = f_\theta(\widetilde{A^2}, \widetilde{X^2})$. To facilitate subsequent discussion, Z^1 and Z^2 are further batch-normalized so that each
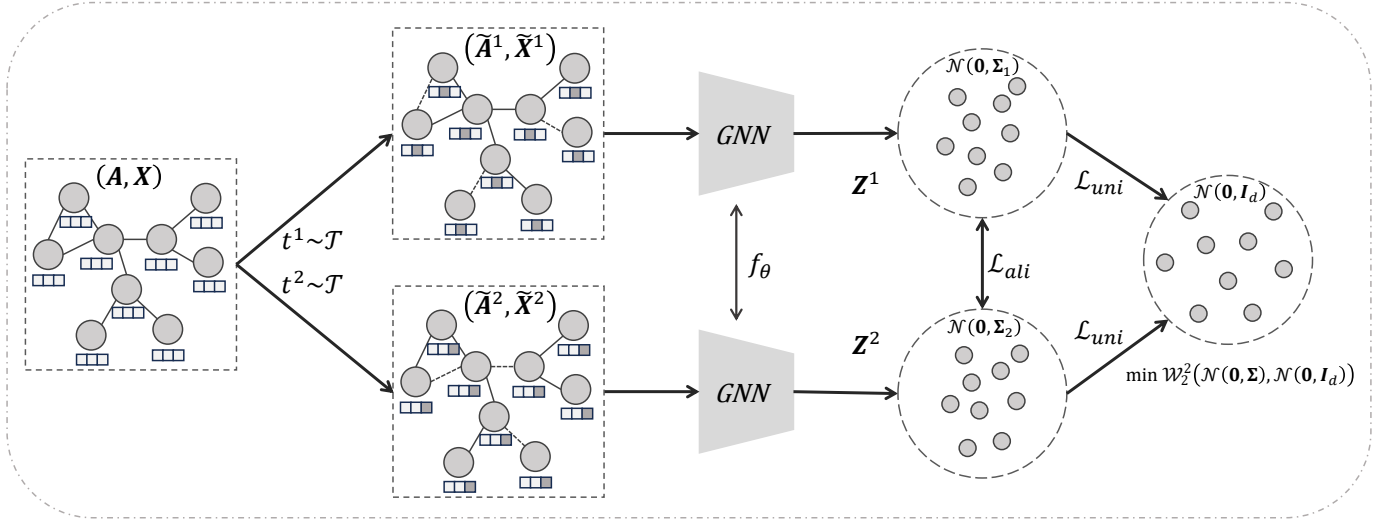
Figure 1: Overview of our proposed graph self-supervised learning framework SSGE. For a given attributed graph (A, X), we first generate two distinct views $(\widetilde{A^1}, \widetilde{X^1})$, $(\widetilde{A^2}, \widetilde{X^2})$ through random augmentations t^1, t^2. These two views are subsequently fed into a shared GNN encoder f_θ to extract batch-normalized node representations Z^1, Z^2. The *alignment* loss $\mathcal{L}_{ali}$ and the *uniformity* loss $\mathcal{L}_{uni}$ are applied on Z^1, Z^2. Here, $\mathcal{L}_{uni}$ aims to minimize the 2-Wasserstein distance between the distribution of learned representations and the isotropic Gaussian distribution $\mathcal{N}(0, I_d)$.

representation channel in which obeys a distribution with zero-mean and one-standard deviation. Finally, the model is optimized using some self-supervised objectives, such as InfoNCE [34] or our proposed objective defined in Eq. (15). After training, to obtain node representations for downstream tasks, the original graph $\mathcal{G} = (A, X)$ is fed into the trained encoder f_θ, yielding $Z = f_\theta(A, X)$.

4.2. Negative-Free Self-Supervised Loss

In this subsection, we first analyze the weakness of contrastive-based self-supervised methods and then propose a negative-free self-supervised loss.

4.2.1. Weakness of Graph Contrastive Learning

In most graph contrastive learning methods [4, 11, 12, 13], both positive pairs and negative pairs are required for learning a model. For instance, the widely adopted InfoNCE [34] loss has the following formulation:

$$\mathcal{L}_{InfoNCE} = \sum_{i=1}^{n} \underbrace{-z_i^1 \cdot z_i^2 / \tau}_{\text{alignment}} + \underbrace{\log\left(\sum_j e^{z_i^1 \cdot z_j^2 / \tau}\right)}_{\text{uniformity}}, \quad (4)$$

where $z_i^1 \in \mathbb{R}^d$ and $z_i^2 \in \mathbb{R}^d$ are the (ℓ_2-normalized) representations of two views of the same sample i, and τ is the temperature hyperparameter. Minimizing Eq. (4) is equivalent to maximizing the cosine similarity of two views of the same sample (*alignment*) and meanwhile minimizing the cosine similarity of two views of different samples (*uniformity*). Intuitively, the *alignment* term makes the positive pairs close to each other, while the *uniformity* term distributes all samples roughly uniformly on the hypersphere $\mathcal{S}^{d-1}$ [16]. Based on this analysis,

Wang and Isola [16] propose a uniformity metric by utilizing the logarithm of the average pairwise Gaussian potential:

$$\mathcal{L}_{uniform} = \log \frac{1}{n(n-1)/2} \sum_{i=2}^{n} \sum_{j=1}^{i-1} e^{(-t\|z_i - z_j\|_2^2)}, t > 0. \quad (5)$$

This uniformity metric is expected to be both asymptotically correct (i.e., the distribution optimizing this metric should converge to uniform distribution) and empirically reasonable with a finite number of samples. However, both the uniformity term in InfoNCE and this pairwise Gaussian potential based uniformity metric inherently rely on the large number and high quality of negative samples. This reliance results in substantial computational and memory overhead, as well as class collision, where diverse samples from the same class are inaccurately deemed negative pairs, thereby impeding representation learning for classification [18, 19]. In this work, we introduce a new negative-free uniformity objective derived from hyperspherical uniform distribution.

4.2.2. Uniformity from Isotropic Gaussian Distribution

We first show that zero-mean isotropic (equal-variance) Gaussian distributed vectors (after normalized to norm 1) are uniformly distributed over the unit hypersphere with the following theorem.

Theorem 1 (Hyperspherical Uniformity [35]). *The normalized vector of Gaussian variables is uniformly distributed on the hypersphere. Formally, let $z_1, z_2, \cdots, z_d \sim \mathcal{N}(0, 1)$ and be independent. Then the vector*

$$z = \left[\frac{z_1}{r}, \frac{z_2}{r}, \cdots, \frac{z_d}{r}\right] \quad (6)$$

follows the uniform distribution on $\mathcal{S}^{d-1}$, where $r = \sqrt{z_1^2 + z_2^2 + \cdots + z_d^2}$ is a normalization factor.

Proof. A random variable has distribution $\mathcal{N}(0,1)$ if it has the density function

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}. \tag{7}$$

A d-dimensional random vector z has distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ if the components are independent and have distribution $\mathcal{N}(0,1)$ each. Then the density of z is given by

$$f(z) = \frac{1}{(\sqrt{2\pi})^d} e^{-\frac{1}{2}\langle z,z \rangle}, \tag{8}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product. Then we introduce the following lemma (Lemma 1) about the orthogonal-invariance of the Gaussian distribution.

Lemma 1. *Let z be a d-dimensional random vector with distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ and $\mathbf{U} \in \mathbb{R}^{d \times d}$ be an orthogonal matrix ($\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}_d$). Then $\mathbf{Y} = \mathbf{U}z$ also has the distribution of $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$.*

Proof. For any measurable set $\mathcal{A} \subset \mathbb{R}^d$, we have that

$$\begin{aligned}
P(\mathbf{Y} \in \mathcal{A}) &= P(\mathbf{Z} \in \mathbf{U}^\top \mathcal{A}) \\
&= \int_{\mathbf{U}^\top \mathcal{A}} \frac{1}{(\sqrt{2\pi})^d} e^{-\frac{1}{2}\langle z,z \rangle} \\
&= \int_{\mathcal{A}} \frac{1}{(\sqrt{2\pi})^d} e^{-\frac{1}{2}\langle \mathbf{U}z, \mathbf{U}z \rangle} \\
&= \int_{\mathcal{A}} \frac{1}{(\sqrt{2\pi})^d} e^{-\frac{1}{2}\langle z,z \rangle}
\end{aligned} \tag{9}$$

because of orthogonality of $\mathbf{U}$. Therefore the lemma holds. $\square$

Since any rotation is just a multiplication with some orthogonal matrix, we know that normally distributed random vectors are invariant to rotation. As a result, generating $z \in \mathbb{R}^d$ with distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ and then projecting it onto the hypersphere $\mathcal{S}^{d-1}$ produces random vectors $\frac{z}{\|z\|_2}$ that are uniformly distributed on the hypersphere. Therefore the theorem holds. $\square$

The aforementioned theorem establishes an equivalence between the hyperspherical uniform distribution and the normalized isotropic Gaussian distribution. Consequently, we can utilize the distance between the distribution of learned representations and the isotropic Gaussian distribution as a measure of uniformity. Specifically, given learned representations $\mathbf{Z} \in \mathbb{R}^{n \times d}$, we first apply batch normalization to ensure that each representation channel follows a distribution with zero mean and unit variance. To simplify computation, we adopt a Gaussian hypothesis for the normalized representations and assume they conform to $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$, where $\mathbf{\Sigma} = \frac{1}{n-1}\mathbf{Z}^\top\mathbf{Z}$, and the on-diagonal elements are 1. Building upon this assumption, we employ the 2-Wasserstein distance, as defined in Eq. (3), to quantify the disparity between the learned representation distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ and the target isotropic Gaussian distribution

$\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ as the uniformity objective:

$$\begin{aligned}
\mathcal{U}(\mathbf{Z}) &= \mathcal{W}_2^2 \left(\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}), \mathcal{N}(\mathbf{0}, \mathbf{I}_d) \right) \\
&= \|\mathbf{0} - \mathbf{0}\|_2^2 + \mathrm{Tr}\left(\mathbf{\Sigma} + \mathbf{I}_d - 2\left(\mathbf{I}_d^{\frac{1}{2}} \mathbf{\Sigma} \mathbf{I}_d^{\frac{1}{2}} \right)^{\frac{1}{2}} \right) \\
&= 2d - 2\,\mathrm{Tr}\left(\mathbf{\Sigma}^{\frac{1}{2}} \right).
\end{aligned} \tag{10}$$

Minimizing this uniformity objective promotes the proximity of the learned representation distribution to the isotropic Gaussian distribution, thereby enhancing the uniformity of the learned representations. Consequently, for the representations $\mathbf{Z}^1$ and $\mathbf{Z}^2$ associated with the two generated graph views during training, the expression for the *uniformity* loss is:

$$\begin{aligned}
\mathcal{L}_{uni} &= \frac{1}{2}(\mathcal{U}(\mathbf{Z}^1) + \mathcal{U}(\mathbf{Z}^2)). \\
&= 2d - \mathrm{Tr}\left(\mathbf{\Sigma}_1^{\frac{1}{2}} \right) - \mathrm{Tr}\left(\mathbf{\Sigma}_2^{\frac{1}{2}} \right),
\end{aligned} \tag{11}$$

where $\mathbf{\Sigma}_1 = \frac{1}{n-1}\mathbf{Z}^{1\top}\mathbf{Z}^1$ and $\mathbf{\Sigma}_2 = \frac{1}{n-1}\mathbf{Z}^{2\top}\mathbf{Z}^2$ are estimated covariance matrices of the two view representations.

4.2.3. Combining with the View Consistency Prior

Similar to the alignment term in contrastive learning objectives, we try to maximize the correlation between two views by minimizing the Euclidean distance between representations derived from one sample:

$$\mathcal{L}_{ali} = \frac{1}{n}\|\mathbf{Z}^1 - \mathbf{Z}^2\|_2^2. \tag{12}$$

Since $\mathbf{Z}^1$ and $\mathbf{Z}^2$ are bath-normalized, $\|\mathbf{Z}^1 - \mathbf{Z}^2\|_2^2$ can be computed by:

$$\begin{aligned}
\|\mathbf{Z}^1 - \mathbf{Z}^2\|_2^2 &= \sum_j^d \sum_i^n \left(z_{i,j}^1 - z_{i,j}^2 \right)^2 \\
&= \sum_j^d \sum_i^n \left(\left(z_{i,j}^1 \right)^2 + \left(z_{i,j}^2 \right)^2 - 2z_{i,j}^1 z_{i,j}^2 \right) \\
&= \sum_j^d \left(2 - 2\left(\mathbf{Z}_{\cdot,j}^1 \right)^\top \mathbf{Z}_{\cdot,j}^2 \right) \\
&= \left(2d - 2\,\mathrm{Tr}\left(\mathbf{Z}^{1\top}\mathbf{Z}^2 \right) \right).
\end{aligned} \tag{13}$$

For simplicity, we adopt the following equivalent form of Eq. (12) as the implementation of our *alignment* loss:

$$\mathcal{L}_{ali} = \frac{1}{n}\mathrm{Tr}\left(\mathbf{Z}^{1\top}\mathbf{Z}^2 \right). \tag{14}$$

The alignment loss promotes the mapping of different augmentation views of the same sample to nearby representations, thus enabling the model to learn representations invariant to unneeded noise factors.

4.2.4. Overall Objective

Combining the *uniformity* loss and the *alignment* loss, we formulate our overall objective as follows:

$$\mathcal{L} = \mathcal{L}_{ali} + \lambda \mathcal{L}_{uni}, \tag{15}$$

where λ is a non-negative hyperparameter that balances the contributions of the two terms.

Complexity Analysis. Consider a graph with n nodes, and each node is embedded into a d-dimensional vector. The computation of the alignment term $\mathcal{L}_{ali}$ requires $O(n)$ time and $O(n)$ space. The computation of $\Sigma^{\frac{1}{2}}$ in the uniformity term $\mathcal{L}_{uni}$ is implemented using eigenvalue decomposition, which takes $O(d^3)$ time and $O(d^2)$ space, with typically $d \ll n$. In contrast, contrastive learning methods [4, 11, 13, 5] treat two views of the same node as positive pairs and views of different nodes as negative pairs, incurring $O(n^2)$ time and $O(n^2)$ space. As a result, our negative-free method holds more promise for handling large-scale graphs without incurring prohibitively high time and space costs compared to contrastive learning methods.

Algorithm 1 The overall procedure of SSGE
___
Input: $\mathcal{G} = (A, X)$
Parameter: Trade-off λ, Augmentor $\mathcal{T}$
Output: The graph encoder f_θ
  1: Initialize model parameters;
  2: **while** not converge **do**
  3:      Sample two augmentation functions $t^1, t^2 \sim \mathcal{T}$;
  4:      Generate two augmented graphs via $t^1(A, X)$ and $t^2(A, X)$;
  5:      Obtain batch-normalized node representations $\mathbf{Z}^1$ and $\mathbf{Z}^2$ using f_θ;
  6:      Compute the uniformity loss via Eq. 11;
  7:      Compute the alignment loss via Eq. 14;
  8:      Update the parameters of f_θ by minimizing Eq. 15;
  9: **end while**
 10: **return** f_θ.

Advantages over Peer Works Here we conduct a systematic comparison with previous graph self-supervised learning methods, including DGI [3], MVGRL [23], GRACE [4], GCA [11], BGRL [19], AFGRL [22], CCA-SSG [20], and G-BT [21]. In brief, our SSGE stands out by mitigating reliance on negative samples and intricate components. Specifically, DGI and MV-GRL require a parameterized Jensen-Shannon estimator for approximating mutual information between two views. MVGRL also introduces asymmetric architectures by employing two different GNNs for the input graph and the diffusion graph, respectively. On the other hand, GRACE and GCA employ an additional MLP-projector followed by an InfoNCE mutual information estimator, which relies on a substantial number of negative samples, leading to heavy computation, memory overhead, and class collision. BGRL and AFGRL require an asymmetric encoder architecture that incorporates Exponential Moving Average (EMA), Stop-Gradient, and an additional projector. CCA-SSG and G-BT may demonstrate suboptimal performance on datasets with smaller feature dimensions, as they essentially perform dimension reduction. In contrast, our framework requires no additional components except a single GNN encoder, our Gaussian distribution-guided objective is negative-free, and our model can also work well on low-dimensional datasets.

4.2.5. In-depth Analysis

We recognize that the optimal solution for the final loss functions of CCA-SSG, G-BT, and our SSGE is $\Sigma = I_d$. Despite such similarity, we demonstrate the superiority of our objective function theoretically as follows.

We begin by briefly discussing the phenomenon of dimensional collapse in representation learning. Dimensional collapse occurs when the embedding occupies a subspace rather than the entire embedding space, which is indicated by one or more zero singular values of the embedding matrix (in other words, one or more zero eigenvalues λ_i of the covariance matrix Σ) [28]. To address dimensional collapse, G-BT, CCA-SSG, and our proposed SSGE decorrelate the covariance matrix through a regularization term. Specifically, G-BT and CCA-SSG minimize $\|\Sigma - I_d\|_F^2$ to encourage the covariance matrix to approximate the identity matrix. In contrast, our SSGE minimizes the distance between the distribution of learned representations and the isotropic Gaussian distribution, which can be simplified as $-\text{Tr}(\Sigma^{\frac{1}{2}})$ according to Eq. (10).

Note that $\|\Sigma - I_d\|_F^2 = \sum_{i=1}^d (\lambda_i - 1)^2$, $-\text{Tr}(\Sigma^{1/2}) = -\sum_{i=1}^d \sqrt{\lambda_i}$, and $\sum_{i=1}^d \lambda_i = \|\mathbf{Z}\|_F^2 = d$ since $\mathbf{Z}$ is batch-normalized. We can reformulate these regularization terms using eigenvalues λ_i: for G-BT and CCA-SSG, the problem becomes $\min \sum_{i=1}^d (\lambda_i - 1)^2$ subject to $\sum_{i=1}^d \lambda_i = d$, while for SSGE, it is $\min -\sum_{i=1}^d \sqrt{\lambda_i}$ under the same constraint. The optimal solution for both optimization problems is $\lambda_i = 1$ for each i.

However, when optimizing model parameters using SGD, G-BT and CCA-SSG may inadequately regularize zero eigenvalues compared to our SSGE. Specifically, the gradient of $\|\Sigma - I_d\|_F^2 = \sum_{i=1}^d (\lambda_i - 1)^2$ with respect to the eigenvalue λ_i is $\nabla_{\lambda_i} = 2(\lambda_i - 1)$. This regularization term imposes a finite penalty on singular values λ_i approaching 0, potentially leading to dimensional collapse if other loss terms offer greater rewards than the penalty.

In contrast, the gradient of $-\text{Tr}(\Sigma^{1/2}) = -\sum_{i=1}^d \sqrt{\lambda_i}$ with respect to λ_i is $\nabla_{\lambda_i} = -\frac{1}{2\sqrt{\lambda_i}}$, resulting in a significantly larger gradient as λ_i approaches 0. Thus, the optimization process will naturally steer away from nearly infinite penalties toward full rank Σ, avoiding dimensional collapse more effectively.

5. Experiments

In this section, we design the experiments to evaluate our proposed SSGE and answer the following research questions. **RQ1:** Does SSGE outperform existing baseline methods on node classification and node clustering? **RQ2:** Is SSGE more efficient than graph contrastive learning baselines? **RQ3:** How does each component of SSGE benefit the performance? **RQ4:** How to intuitively understand the role of our proposed uniformity objective? **RQ5:** Are learned node representations uniformly distributed on the hypersphere? **RQ6:** Is SSGE sensitive to hyperparameters?

5.1. Experiment Setup

5.1.1. Datasets

We adopt seven publicly available real-world benchmark datasets, including three citation networks Cora, CiteSeer, and PubMed, one reference network WikiCS, one co-purchase network Computer, one co-authorship network CoauthorCS, and one large-scale citation network ArXiv to conduct the experiments throughout the paper. The statistics of the datasets are provided in Table 1. We give their detailed descriptions as follows:

Table 1: Dataset statistics.

Dataset	Type	#Nodes	#Edges	#Feats	#Cls
Cora	citation	2,708	10,556	1,433	7
CiteSeer	citation	3,327	9,228	3,703	6
PubMed	citation	19,717	88,651	500	3
WikiCS	reference	11,701	431,726	300	10
Computer	co-purchase	13,752	491,722	767	10
CoauthorCS	co-authorship	18,333	163,788	6,805	15
ArXiv	citation	169,343	2,315,598	128	40

- **Cora**, **CiteSeer**, **PubMed** [32] are three well-known citation network datasets, in which nodes represent publications and edges indicate their citations. Each node in Cora and CiteSeer is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary, while each node in PubMed is described by a TF/IDF weighted word vector from the dictionary. All nodes are labeled based on the respective paper subjects.

- **WikiCS** [36] is a reference network constructed from Wikipedia. It comprises nodes corresponding to articles in the field of Computer Science, where edges are derived from hyperlinks. The dataset includes 10 distinct classes representing various branches within the field. The node features are computed as the average GloVe word embeddings of the respective articles.

- **Computer** [37] is a network constructed from Amazon's co-purchase relationships. Nodes represent goods, and edges indicate frequent co-purchases between goods. The node features are represented by bag-of-words encoding of product reviews, and class labels are assigned based on the respective product categories.

- **CoauthorCS** [37] is a co-authorship network based on the Microsoft Academic Graph. Here, nodes are authors, that are connected by an edge if they co-authored a paper; node features represent paper keywords for each author's papers, and class labels indicate the most active fields of study for each author.

- **ArXiv** [37] is a citation network between most Computer Science arXiv papers indexed by Microsoft Academic Graph, where nodes represent papers and edges represent citation relations. Each node is described by a 128-dimensional feature vector obtained by averaging the skip-gram word embeddings in its title and abstract. The nodes are categorized by their related research area.

5.1.2. Baselines

To verify the effectiveness of our proposed model, our evaluation includes a comprehensive comparison of SSGE with 22 baseline methods. We divide all baseline models into the following four categories:

- Supervised methods: MLP and graph neural networks GCN [32] and GAT [38];

- Traditional unsupervised models: random-walk based graph embedding methods DeepWalk [39] and Node2Vec [40], graph auto-encoders GAE and VGAE [41];

- Contrastive learning models: GMI [26], DGI [3], GGD [42], MVGRL [23], GRACE [4], GCA [11], MERIT [43], gCooL [12], COSTA [13], and HomoGCL [5];

- Non-contrastive self-supervised learning models: BGRL [19], AFGRL [22], RGRL [29], CCA-SSG [20], and G-BT [21].

Table 2: Hyperparameter Specifications.

Dataset	p_d, p_m	lr, wd	λ	#hid_units	#epochs
Cora	0.3, 0.1	1e-3, 1e-5	0.1	256-256	80
CiteSeer	0.4, 0.0	1e-3, 1e-5	0.05	512	20
PubMed	0.3, 0.5	1e-3, 1e-5	0.6	512-256	100
WikiCS	0.8, 0.1	1e-2, 1e-6	0.5	256-256	50
Computer	0.1, 0.3	1e-3, 1e-5	1.0	512-512	120
CoauthorCS	1.0, 0.2	1e-3, 1e-5	0.05	512-512	80
ArXiv	0.5, 0.3	1e-2, 1e-6	3.0	512-512	400

5.1.3. Evaluation Protocol

We evaluate the performance of SSGE on two downstream tasks, i.e., node classification and node clustering. Following previous work [5], We first train the model in an unsupervised manner. Then we freeze the parameters of the encoder to generate node representations for downstream tasks. For node classification, we use the learned representations to train and test a simple logistic regression classifier on public splits for Cora, CiteSeer, PubMed, WikiCS, and ArXiv, and ten 1:1:8 train/validation/test random splits for Computer and CoauthorCS, as they have no publicly accessible splits. We train the model for ten runs and report the performance in terms of accuracy. For node clustering, we train a k-means model on the learned representations ten times, where the number of clusters is set to the number of classes for each dataset. We measure the clustering

Table 3: Node classification results measured by accuracy along with standard deviations. The *Input* column illustrates the data used in the training stage, and Y denotes labels. '-' means the method is out-of-memory under a full-graph training setting.

	Method	Input	Cora	CiteSeer	PubMed	WikiCS	Computer	CoauthorCS	ArXiv
Supervised	MLP	X, Y	57.8±0.2	54.2±0.1	72.8±0.2	71.98±0.00	73.81±0.00	90.37±0.00	55.50±0.23
	GCN	A, X, Y	81.5±0.4	70.2±0.4	79.0±0.2	77.19±0.12	86.51±0.54	93.03±0.31	71.74±0.29
	GAT	A, X, Y	83.0±0.7	72.5±0.7	79.0±0.3	77.65±0.11	86.93±0.29	92.31±0.24	72.10±0.13
Traditional	DeepWalk	A	68.5±0.5	49.8±0.2	66.2±0.7	74.35±0.06	85.68±0.06	84.61±0.22	
	Node2Vec	A	70.1±0.4	49.8±0.3	69.8±0.7	71.79±0.05	84.39±0.08	85.08±0.03	70.07±0.13
	GAE	A, X	71.5±0.4	65.8±0.4	72.1±0.5	77.87±0.53	85.27±0.19	90.01±0.71	-
	VGAE	A, X	73.0±0.3	68.3±0.4	75.8±0.2	77.87±0.53	86.37±0.21	92.11±0.09	-
Contrastive	GMI	A, X	83.0±0.3	72.4±0.1	79.9±0.2	74.85±0.08	82.21±0.31	88.78±0.12	-
	DGI	A, X	82.3±0.6	71.8±0.7	76.8±0.6	78.25±0.56	83.95±0.47	92.15±0.63	70.34±0.16
	GGD	A, X	83.5±0.6	73.0±0.6	81.0±0.8	78.62±0.47	88.12±0.62	92.30±0.23	71.20±0.20
	MVGRL	A, X	83.5±0.4	73.3±0.5	80.1±0.7	77.57±0.46	87.52±0.11	92.11±0.12	-
	GRACE	A, X	81.9±0.4	71.3±0.3	80.1±0.2	78.64±0.33	88.29±0.11	92.17±0.04	-
	GCA	A, X	81.7±0.3	71.1±0.4	79.5±0.5	78.35±0.05	87.85±0.31	93.10±0.01	-
	MERIT	A, X	83.1±0.6	73.5±0.7	80.1±0.4	78.35±0.05	88.01±0.12	92.51±0.14	-
	gCooL	A, X	82.8±0.5	72.0±0.3	80.2±0.4	78.74±0.04	88.67±0.10	92.75±0.01	-
	COSTA	A, X	82.2±0.2	70.7±0.5	80.4±0.3	78.82±0.12	88.32±0.03	92.94±0.10	-
	HomoGCL	A, X	**84.1±0.5**	72.3±0.7	81.1±0.3	78.26±0.32	88.46±0.20	92.28±0.03	-
Non-contrastive	BGRL	A, X	82.7±0.6	71.1±0.8	79.6±0.5	78.41±0.09	87.89±0.10	92.72±0.03	71.44±0.12
	AFGRL	A, X	79.8±0.2	69.4±0.2	80.0±0.1	77.62±0.49	88.12±0.27	93.07±0.17	-
	RGRL	A, X	83.5±0.7	71.5±0.9	79.8±0.3	78.78±0.64	88.45±0.52	92.94±0.13	71.49±0.08
	CCA-SSG	A, X	83.9±0.4	73.0±0.3	80.7±0.4	77.92±0.65	88.76±0.36	93.01±0.20	71.21±0.20
	G-BT	A, X	83.6±0.0	72.9±0.4	80.4±0.1	76.83±0.73	87.93±0.36	92.91±0.25	71.12±0.18
	SSGE	A, X	83.9±0.3	**74.1±0.3**	**81.6±0.1**	**79.18±0.57**	**89.05±0.58**	**93.46±0.45**	71.62±0.19

performance in terms of two prevalent metrics Normalized Mutual Information (NMI) score: NMI = $2I(\hat{Y}; Y)/[H(\hat{Y}) + H(Y)]$, where $\hat{Y}$ and Y being the predicted cluster indexes and class labels respectively, $I(\cdot)$ being the mutual information, and $H(\cdot)$ being the entropy; and Adjusted Rand Index (ARI): ARI = RI $- \mathbb{E}[\text{RI}]/(\max\{\text{RI}\} - \mathbb{E}[\text{RI}])$, where RI being the Rand Index.

5.1.4. Implementation Details

We implement our model with PyTorch. All experiments are conducted on a V100 GPU with 32 GB of memory. The graph encoder f_θ is specified as a standard two-layer GCN model with the ELU activation [44] for all the datasets except CiteSeer (where we empirically find that a one-layer GCN is better). During training, we employ the Adam SGD optimizer [45] with a learning rate and weight decay of either (1e-3, 1e-5) or (1e-2, 1e-6). The augmentation function set $\mathcal{T}$ is controlled by the edge dropping ratio p_d and the feature masking ratio p_m. We utilize the processed versions of all datasets provided by the Deep Graph Library [46]. Detailed hyperparameters can be found in Table 2. The implementation code is available at `https://github.com/Cloudy1225/SSGE`.

5.2. Main Results

5.2.1. Performance Comparison (**RQ1**)

The experimental results of node classification on seven datasets are shown in Table 3. As we can see, SSGE outperforms all self-supervised baselines on six out of seven datasets, despite its simple architecture. On Cora, SSGE achieves competitive results as the most powerful baseline HomoGCL. It is worth mentioning that we empirically find that on CoauthorCS, a pure two-layer MLP encoder is better than GNN models. This might be because the graph-structured information is much less informative than the node features, presumably providing harmful signals for classification. We also evaluate the node clustering performance on the three citation networks Cora, CiteSeer, and PubMed. As shown in Table 4, SSGE can always yield significant improvements over other methods, especially on Cora with a 2.7% gain for NMI and a 3.3% gain for ARI. These results clearly demonstrate the effectiveness of our method.

5.2.2. Efficiency Comparison (**RQ2**)

We conduct a comparative analysis with several graph contrastive learning methods regarding the number of model parameters, training time, and memory costs on datasets including Cora, CiteSeer, PubMed, and Computer. Table 5 summarizes all indicators of various methods. Overall, SSGE has fewer parameters, shorter training time, and fewer memory costs in

Table 4: Node clustering results measured by NMI and ARI along with standard deviations.

Dataset	Cora		CiteSeer		PubMed	
Metric	NMI	ARI	NMI	ARI	NMI	ARI
k-means	15.44±3.83	9.49±2.01	20.66±2.83	16.80±3.02	31.34±0.15	28.12±0.03
SC	46.07±0.99	34.39±0.98	23.95±0.53	18.48±0.42	28.75±0.00	30.34±0.00
VGAE	52.48±1.33	43.99±2.34	34.46±0.92	32.65±0.92	27.16±1.45	26.32±1.15
DGI	55.82±0.60	48.91±1.42	41.16±0.54	39.78±0.74	25.27±0.02	24.06±0.03
MVGRL	56.30±0.27	50.28±0.40	43.47±0.08	44.09±0.09	27.07±0.00	24.53±0.00
GRACE	55.82±0.60	48.91±1.42	39.07±0.07	40.38±0.08	30.44±0.02	30.62±0.01
GCA	52.82±1.11	46.29±1.80	41.08±0.28	41.72±0.30	31.62±0.08	30.76±0.17
gCooL	50.25±1.08	44.95±1.74	41.67±0.41	42.66±0.47	33.14±0.02	31.93±0.01
HomoGCL	57.87±1.47	53.68±1.77	40.32±0.07	40.10±0.09	27.67±1.78	25.59±0.84
BGRL	55.38±0.49	47.10±0.35	38.95±0.33	38.81±0.12	28.43±0.10	24.81±0.08
CCA-SSG	56.51±1.49	50.77±3.39	43.69±0.24	44.26±0.23	29.61±0.01	25.81±0.01
G-BT	55.54±1.19	48.39±2.44	42.78±0.25	43.89±0.21	30.12±0.03	29.32±0.02
SSGE	**60.58±0.25**	**56.96±0.34**	**45.27±0.33**	**46.87±0.37**	**33.42±0.12**	**32.05±0.11**

Table 5: Comparison of the number of parameters, training time for achieving the best performance, and the memory cost of different methods on Cora, CiteSeer, PubMed, and Computer.

Dataset	Cora			CiteSeer			PubMed			Computer		
	#Paras	Time	Mem	#Paras	Time	Mem	#Paras	Time	Mem	#Paras	Time	Mem
DGI	1,260K	4.06s	570MB	2,422K	4.51s	710MB	782K	10.24s	1,024MB	919K	13.50s	926MB
MVGRL	1,731K	17.73s	3,838MB	4,055K	15.48s	7,386MB	775K	83.64s	4,622MB	1,049K	124.96s	4,660MB
GRACE	564K	10.57s	748MB	2,159K	7.59s	1,018MB	326K	396.97s	12,744MB	394K	267.72s	6,630MB
HomoGCL	866K	3.95s	1,014MB	2,028K	6.01s	1,478MB	388K	167.97s	22,010MB	525K	92.35s	11,788MB
SSGE	433K	3.41s	580MB	1,896K	2.44s	800MB	388K	7.90s	1,256MB	656K	15.42s	1,234MB

most cases. This is because our method does not rely on additional projection heads, parameterized mutual information estimator, and negative samples, which contribute to increased computational load, additional parameters, and storage requirements. Besides, the short training time potentially indicates the fast convergence of our algorithm. Despite its simplicity and efficiency, our method achieves even better (or competitive) performance.

5.3. In-Depth Analysis

5.3.1. Effectiveness of Alignment/Uniformity Terms (RQ3)

To verify the effects of each loss component, we conduct ablation studies with varying combinations of the alignment term and the uniformity term. We assess their impact on node classification across four datasets, and the results are presented in Table 6. It is observed that only using the alignment term will lead to a performance drop instead of completely collapsed solutions since node representations are batch-normalized to have zero-mean and one-standard deviation. On the other hand, optimizing only the uniformity term yields unsatisfactory results, as the model learns nothing meaningful but Gaussian distributed

representations. These results highlight the effectiveness of incorporating the alignment term and the uniformity term to learn discriminative node representations.

Variants	Cora	CiteSeer	PubMed	WikiCS
$\mathcal{L}_{ali}$	79.7±0.1	71.9±0.2	78.1±0.2	77.02±0.40
$\mathcal{L}_{uni}$	48.7±0.6	30.1±0.7	51.4±0.5	76.47±0.60
$\mathcal{L}_{ali} + \mathcal{L}_{uni}$	83.9±0.3	74.1±0.3	81.6±0.1	79.18±0.57

Table 6: Ablation study of node classification accuracy (%) on the key components of SSGE.

5.3.2. Visualization of Feature Covariance Matrix (RQ4)

To gain a visual insight into the role of the uniformity term, we present visualizations depicting the normalized covariance matrix Σ of learned representations under various settings on Cora. As depicted in Figure 2(a), the off-diagonal elements of the covariance matrix approach 1 when the uniformity term is not considered. This suggests that various dimensions of the

representation matrix are coupled together, signifying the occurrence of dimensional collapse (all the embeddings lie in a line). From Figure 2(b) and Figure 2(c), we observe that the proposed uniformity objective effectively decorrelates diverse representation dimensions, thereby mitigating the issue of dimensional collapse.
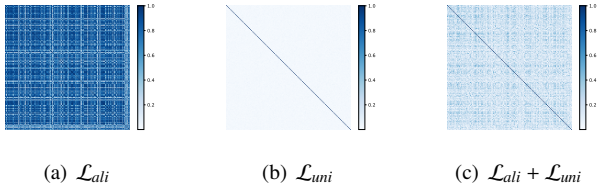

(a) $\mathcal{L}_{ali}$ (b) $\mathcal{L}_{uni}$ (c) $\mathcal{L}_{ali} + \mathcal{L}_{uni}$

Figure 2: Visualization of the covariance matrix (absolute value) of learned representations on Cora.

5.3.3. Visualization of Feature Distribution on $\mathcal{S}^1$ (RQ5)

To verify the uniformity of node representations learned by SSGE, we visualize feature distributions of all classes and selected specific classes on $\mathcal{S}^1$ using Gaussian kernel density estimation in $\mathbb{R}^2$. Figure 3 and Figure 4 summarize the resulting distributions of learned representations on Cora and CiteSeer, respectively. As can be seen, representations learned by SSGE are both aligned (having low intra-class distances) and uniform (evenly distributed on $\mathcal{S}^1$).
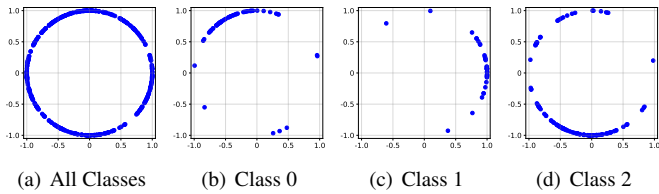

(a) All Classes (b) Class 0 (c) Class 1 (d) Class 2

Figure 3: Visualizing the alignment and uniformity of node representations on Cora.

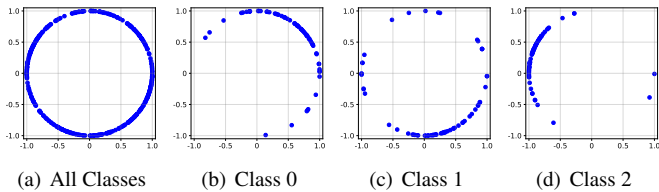(a) All Classes (b) Class 0 (c) Class 1 (d) Class 2

Figure 4: Visualizing the alignment and uniformity of node representations on CiteSeer.

5.4. Hyperparameter Analysis (RQ6)

5.4.1. Impact of Uniformity Intensity

We investigate how the intensity of uniformity influences the performance as the trade-off hyperparameter λ is varied. Figure 5(a) depicts the variation in classification accuracy with different values of λ across Cora, WikiCS, Computer, and CoauthorCS. It can be observed that at the beginning, increasing λ

enhances performance, but excessively large values of λ result in a significant degradation of performance. When λ is too small, the uniformity term fails to fully leverage its role in promoting uniform representations. Conversely, when λ is excessively large, placing too much emphasis on uniformity while neglecting alignment leads to meaningless representations. To choose the value of λ, we recommend conducting a grid search or a sensitivity analysis on a subset of datasets during the initial experimentation phase. This approach can help identify a suitable range for λ that balances performance across different contexts.
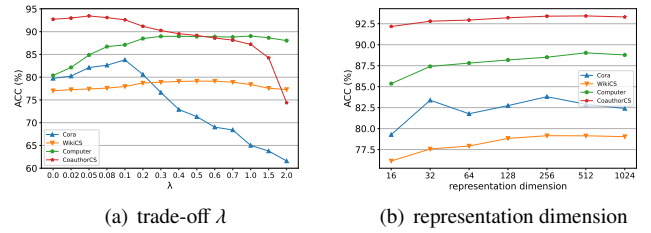

(a) trade-off λ (b) representation dimension

Figure 5: Impact of the trade-off λ and representation dimension.

5.4.2. Impact of Representation Dimension

We conduct experiments to explore the impact of varying the representation dimension on performance, as illustrated in Figure 5(a). Similar to other self-supervised methods such as DGI [3], GRACE [4], and BGRL [19], our method achieves optimal performance with an appropriately large dimension (usually 256 or 512), while a too large dimension 1,024 results in a slight decrease in performance. In comparison to feature decorrelation methods like CCA-SSG [20] and G-BT [21], our method can also work well with a representation dimension smaller than the input dimension, on low-dimensional datasets such as WikiCS and PubMed.

5.4.3. Impact of Augmentation Intensity

We further conduct a sensitivity analysis on the augmentation intensity by investigating the effects of different combinations of the feature masking ratio p_m and the edge dropping ratio p_d. The results of node classification on Cora, CiteSeer, and WikiCS are presented in Figure 6. Overall, our method demonstrates robustness to augmentation intensity: within an appropriate range of p_m and p_d, our approach consistently achieves competitive results.
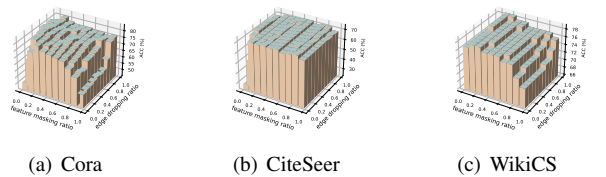

(a) Cora (b) CiteSeer (c) WikiCS

Figure 6: Impact of the augmentation intensity.

6. Conclusion

In this paper, we introduce a negative-free self-supervised objective, drawing inspiration from the fact that the normalized isotropic Gaussian distributed points are uniformly distributed on the unit hypersphere. Our new objective induces a simple and light model without reliance on negative pairs, a parameterized mutual information estimator, an additional projector or asymmetric architectures. Extensive experiments on node classification and node clustering across seven graph benchmarks illustrate that our model achieves competitive performance with fewer parameters, shorter training times, and lower memory costs compared to existing contrastive learning methods.

Limitations and Future Work: Our empirical studies were limited to static homogeneous graphs. Future work will need to explore extensions to other types of graphs, including heterogeneous and dynamic graphs. Additionally, our framework assumes that the learned representations follow a Gaussian distribution for the sake of simplicity. While our framework has been experimentally validated, it may not be the most suitable distribution in all contexts.

Acknowledgments

References

[1] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, P. S. Yu, Graph self-supervised learning: A survey, IEEE Trans. on Knowl. and Data Eng. 35 (6) (2022) 5879–5900.

[2] W. Ju, Z. Fang, Y. Gu, Z. Liu, Q. Long, Z. Qiao, Y. Qin, J. Shen, F. Sun, Z. Xiao, et al., A comprehensive survey on deep graph representation learning, Neural Networks (2024) 106207.

[3] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, R. D. Hjelm, Deep graph infomax, in: International Conference on Learning Representations, 2019.

[4] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Deep graph contrastive representation learning, ArXiv abs/2006.04131 (2020).

[5] W.-Z. Li, C.-D. Wang, H. Xiong, J.-H. Lai, Homogcl: Rethinking homophily in graph contrastive learning, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 1341–1352.

[6] Y. Liu, H. Zhang, T. He, T. Zheng, J. Zhao, Bootstrap latents of nodes and neighbors for graph self-supervised learning, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2024.

[7] H. Liang, X. Du, B. Zhu, Z. Ma, K. Chen, J. Gao, Graph contrastive learning with implicit augmentations, Neural Networks 163 (2023) 156–164.

[8] M. Liu, K. Liang, Y. Zhao, W. Tu, S. Zhou, X. Gan, X. Liu, K. He, Self-supervised temporal graph learning with temporal and structural intensity alignment, IEEE Transactions on Neural Networks and Learning Systems (2024).

[9] M. Liu, Y. Liu, K. LIANG, W. Tu, S. Wang, sihang zhou, X. Liu, Deep temporal graph clustering, in: The Twelfth International Conference on Learning Representations, 2024.

[10] H. Wu, Y. Wu, N. Li, M. Yang, J. Zhang, M. K. Ng, J. Long, High-order proximity and relation analysis for cross-network heterogeneous node classification, Machine Learning (2024) 1–26.

[11] 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, WWW'21, 2021, p. 2069–2080.

[12] B. Li, B. Jing, H. Tong, Graph communal contrastive learning, in: Proceedings of the ACM Web Conference 2022, WWW'22, 2022, p. 1203–1213.

[13] Y. Zhang, H. Zhu, Z. Song, P. Koniusz, I. King, Costa: Covariance-preserving feature augmentation for graph contrastive learning, Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (2022).

[14] H. Duan, C. Xie, B. Li, P. Tang, Self-supervised contrastive graph representation with node and graph augmentation, Neural Networks 167 (2023) 223–232.

[15] Y. Yuan, B. Xu, H. Shen, Q. Cao, K. Cen, W. Zheng, X. Cheng, Towards generalizable graph contrastive learning: An information theory perspective, Neural Networks 172 (2024) 106125.

[16] T. Wang, P. Isola, Understanding contrastive representation learning through alignment and uniformity on the hypersphere, in: Proceedings of the 37th International Conference on Machine Learning, Vol. 119, 2020, pp. 9929–9939.

[17] X. Chen, K. He, Exploring simple siamese representation learning, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 15745–15753.

[18] N. Saunshi, O. Plevrakis, S. Arora, M. Khodak, H. Khandeparkar, A theoretical analysis of contrastive unsupervised representation learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 5628–5637.

[19] S. Thakoor, C. Tallec, M. G. Azar, M. Azabou, E. L. Dyer, R. Munos, P. Veličković, M. Valko, Large-scale representation learning on graphs via bootstrapping, in: International Conference on Learning Representations, 2022.

[20] H. Zhang, Q. Wu, J. Yan, D. Wipf, P. S. Yu, From canonical correlation analysis to self-supervised graph neural networks, in: Advances in Neural Information Processing Systems, Vol. 34, 2021, pp. 76–89.

[21] P. Bielak, T. Kajdanowicz, N. V. Chawla, Graph barlow twins: A self-supervised representation learning framework for graphs, Knowledge-Based Systems 256 (2022) 109631.

[22] N. Lee, J. Lee, C. Park, Augmentation-free self-supervised learning on graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2022, pp. 7372–7380.

[23] K. Hassani, A. H. Khasahmadi, Contrastive multi-view representation learning on graphs, in: Proceedings of the 37th International Conference on Machine Learning, ICML'20, 2020.

[24] K. Cen, H. Shen, Q. Cao, B. Xu, X. Cheng, Towards powerful graph contrastive learning without negative examples, in: 2022 International Joint Conference on Neural Networks (IJCNN), IEEE, 2022, pp. 1–9.

[25] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, Y. Bengio, Learning deep representations by mutual information estimation and maximization, in: International Conference on Learning Representations, 2019.

[26] 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.

[27] R. Yan, P. Bao, X. Zhang, Z. Liu, H. Liu, Towards alignment-uniformity aware representation in graph contrastive learning, in: Proceedings of the 17th ACM International Conference on Web Search and Data Mining, 2024, pp. 873–881.

[28] L. Jing, P. Vincent, Y. LeCun, Y. Tian, Understanding dimensional collapse in contrastive self-supervised learning, in: International Conference on Learning Representations, 2022.
URL https://openreview.net/forum?id=YevsQ05DEN7

[29] N. Lee, D. Hyun, J. Lee, C. Park, Relational self-supervised learning on graphs, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 1054–1063.

[30] A. Ermolov, A. Siarohin, E. Sangineto, N. Sebe, Whitening for self-

supervised representation learning, in: International conference on machine learning, PMLR, 2021, pp. 3015–3024.

[31] S. Nowozin, B. Cseke, R. Tomioka, f-gan: Training generative neural samplers using variational divergence minimization, Advances in neural information processing systems 29 (2016).

[32] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.

[33] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International conference on machine learning, PMLR, 2017, pp. 214–223.

[34] M. Gutmann, A. Hyvärinen, Noise-contrastive estimation: A new estimation principle for unnormalized statistical models, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 297–304.

[35] W. Liu, R. Lin, Z. Liu, L. Xiong, B. Schölkopf, A. Weller, Learning with hyperspherical uniformity, in: International Conference On Artificial Intelligence and Statistics, PMLR, 2021, pp. 1180–1188.

[36] P. Mernyei, C. Cangea, Wiki-cs: A wikipedia-based benchmark for graph neural networks, ArXiv abs/2007.02901 (2020).

[37] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, ArXiv abs/1811.05868 (2018).

[38] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018.

[39] 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.

[40] 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, KDD '16, 2016, p. 855–864. `doi:10.1145/2939672.2939754`.

[41] T. Kipf, M. Welling, Variational graph auto-encoders, ArXiv abs/1611.07308 (2016).

[42] Y. Zheng, S. Pan, V. Lee, Y. Zheng, P. S. Yu, Rethinking and scaling up graph contrastive learning: An extremely efficient approach with group discrimination, Advances in Neural Information Processing Systems 35 (2022) 10809–10820.

[43] M. Jin, Y. Zheng, Y.-F. Li, C. Gong, C. Zhou, S. Pan, Multi-scale contrastive siamese networks for self-supervised graph representation learning, in: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, 2021, pp. 1477–1483. `doi:10.24963/ijcai.2021/204`.

[44] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), arXiv: Learning (2015).

[45] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR abs/1412.6980 (2014).

[46] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, Z. Zhang, Deep graph library: A graph-centric, highly-performant package for graph neural networks, arXiv preprint arXiv:1909.01315 (2019).