Contrastive Laplacian Eigenmaps

Anonymous Author(s)
Affiliation
Address
email

Abstract

Graph contrastive learning attracts/disperses node representations for similar/dissimilar sample pairs under some notion of similarity, resulting from intrinsic properties of data grouping, and it may be combined with a low-dimensional embedding of nodes to preserve intrinsic and structural properties of a graph. In this paper, we extend the celebrated Laplacian Eigenmaps with contrastive learning, and call them COntrastive Laplacian EigenmapS (COLES). Starting from a GAN-inspired contrastive formulation, we show the Jensen-Shannon divergence, which underlies many contrastive graph embeddings, fails under disjoint positive and negative distributions that naturally emerge during contrastive sampling. In contrast, we demonstrate analytically that COLES essentially minimizes a surrogate of Wasserstein distance, and the Wasserstein distance is known to cope well under disjoint distributions. Moreover, we show that the loss of COLES belongs to the family of so-called block-contrastive losses, previously shown to be superior compared to pair-wise losses typically used by contrastive methods. We show on popular benchmarks/backbones that COLES yields favourable accuracy/scalability compared to DeepWalk, GCN, Graph2Gauss, DGI and GRACE baselines.

1 Introduction

Celebrated graph embedding methods, including Laplacian Eigenmaps [3] and IsoMap [30], reduce the dimensionality of the data by assuming that it lies on a low-dimensional manifold. The objective function proposed in studies [3, 30] models the pairwise node similarity [5] by encouraging that embeddings of nodes lie close in the embedding space if the nodes are closely related. In other words, such penalties do not guarantee that unrelated graph nodes are separated from each other in the embedding space. For instance, Elastic Embedding [6] uses data-driven affinities for the so-called local distance term and the data-independent repulsion term.

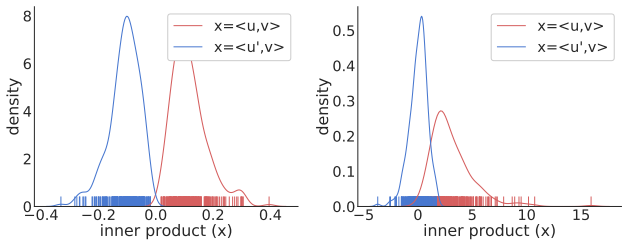
In contrast, modern graph embedding models, often unified under a Sampled Noise Contrastive Estimation (SampledNCE) framework [22, 18] and extended to graph learning [29, 10, 35], enjoy contrastive objectives. By maximizing the mutual information between patch representations and high-level summaries of graph, Deep Graph Infomax (DGI) [31] is a contrastive method. GraphSAGE [10] minimizes/maximizes distances between so-called positive/negative pairs, respectively. It relies on the inner product passed by the sigmoid non-linearity, which we argue below as suboptimal.

In this paper, we propose a new COntrastive Laplacian EigenmapS (COLES) framework for unsupervised network embedding. COLES, derived from SampledNCE framework [22, 18], realizes the negative sampling strategy for Laplacian Eigenmaps. Our general objective is given as:

$$\mathbf{\Theta}^* = \arg\max_{\mathbf{\Theta}} \mathrm{Tr}(f_{\mathbf{\Theta}}(\mathbf{X})^\top \Delta \mathbf{W} f_{\mathbf{\Theta}}(\mathbf{X})) + \beta \Omega(f_{\mathbf{\Theta}}(\mathbf{X})), \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{n \times d}$ denotes the node feature matrix with d feature dimensions given n nodes, $f_{\mathbf{\Theta}}(\mathbf{X}) \in \mathbb{R}^{n \times d'}$ is an output of a chosen Graph Neural Network backbone (embeddings to optimize) with the

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) (b)

Figure 1: Densities of dot-product scores (red and blue curves) between the anchor/positive embedding $\langle \mathbf{v}, \mathbf{u} \rangle$ and the anchor/negative embedding $\langle \mathbf{v}, \mathbf{u}' \rangle$ from GCN (contrastive setting). Left/right figures correspond to two minibatches sampled on Cora. With the small overlap of distributions, traditional contrastive methods relying on the JS divergence may underperform (see Section 4.1 for details).

feature dimension d', $\mathbf{\Theta}$ denotes network parameters, whereas $\Delta\mathbf{W} \in \mathbb{S}_+^n$ is the difference between the degree-normalized positive and negative adjacency matrices representing the data graph and some negative graph capturing negative links for contrastive learning. Moreover, $\beta \geq 0$ controls the regularization term $\Omega(\cdot)$ whose role is to constrain the ℓ_2 norm of network outputs or encourage the so-called incoherence [25] between column vectors. Section 3.1 presents COLES for the Linear Graph Network (LGN) family, in which we take special interest due to their simplicity and agility.

By integrating the analyses of frameworks [18, 33], we show that COLES can be derived by reformulating SampledNCE into Wasserstein GAN using a GAN-inspired contrastive formulation. This result has a profound impact on the performance of COLES, as the standard contrastive approaches based on SampledNCE strategy (*i.e.*, GraphSAGE [10]) turn out to in fact utilize the Jensen-Shannon divergence, which yields zero for disjoint distributions of positive and negative sampled pairs used for contrastive learning. Figure 1 shows two examples of such nearly disjoint distributions. In contrast, COLES by design avoids the sigmoid in favour of the Radial Basis Function (RBF) non-linearity. We show that such a choice coincides with the use of a surrogate of Wasserstein distance (the Wasserstein distance is known for its good properties under poor overlap of distributions), leading to the superiority of COLES. Moreover, we also show that the loss of COLES belongs to the family of so-called block-contrastive losses, which were shown as superior compared to pair-wise losses [27]. Our contributions are threefold:

i. We derive COLES, a reformulation of the Laplacian Eigenmaps into a contrastive setting, based on the SampledNCE framework [22, 18].

ii. By using a formulation inspired by GAN, we show that COLES essentially minimizes a surrogate of Wasserstein distance, as opposed to the use of the Jensen-Shannon (JS) divergence emerging in traditional contrastive methods. Specifically, we show that our formulation enjoys the Kantorovich-Rubinstein duality for the Wasserstein distance, by showing the Lipschitz continuous nature of our formulation.

iii. We show COLES enjoys a block-contrastive loss known to outperform pair-wise losses [27].

Novelty. We propose a simple way to obtain contrastive parametric graph embeddings which works with numerous backbones. For instance, we can obtain spectral graph embeddings by combining COLES with SGC [34] and S^{2}GC [40], which can be solved by the SVD decomposition.

2 Preliminaries

Notations. Let $G = (V, E)$ be a simple, connected and undirected graph with n nodes and m edges. We use $i \in \{1, \cdots, n\}$ to denote the node index of G, and d_j denote the degree of node j in G. Let $\mathbf{W}$ be the adjacency matrix and $\mathbf{D}$ the diagonal matrix containing degrees of nodes. Moreover, let $\mathbf{X} \in \mathbb{R}^{n \times d}$ denote the node feature matrix and each node v is associated with a d-dimensional feature vector $\mathbf{x}_v$. Let the normalized graph Laplacian matrix be defined as $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\widehat{\mathbf{W}}\mathbf{D}^{-1/2} \in \mathbb{S}_+^n$, a symmetric positive semi-definite matrix. Finally, scalars and vectors are denoted by lowercase regular and boldface fonts, respectively, whereas matrices are denoted by uppercase boldface fonts.

2.1 Negative Sampling

SampledNCE [22, 18], a contrastive learning framework, is used by several works [29, 10, 35]. Let $p_d(u|v)$ and $p_n(u'|v)$ be the so-called data and negative distributions given the so-called anchor node v, where u and u' denote the node for a positive and negative sample, respectively. Let $p_d(v)$ be the anchor distribution. For some loss components $s_\Theta(v,u)$ and $\tilde{s}_\Theta(v,u')$ evaluating the similarity for pairs (v,u) and (v,u'), the contrastive loss is typically given as:

$$J(\mathbf{\Theta}) = \mathbb{E}_{v\sim p_d(v)}\left[\mathbb{E}_{u\sim p_d(u|v)} s_\Theta(v,u) + \eta \mathbb{E}_{u'\sim p_n(u'|v)} \tilde{s}_\Theta(v,u')\right], \tag{2}$$

where $\eta > 0$ controls the impact of negative sampling. Let $\mathbf{u} \in \mathbb{R}^{d'}$ be the embedding of the node u obtained with an encoder $f_{\mathbf{\Theta}}(\mathbf{x}_u)$ given parameters $\mathbf{\Theta}$, where $\mathbf{x}_u \in \mathbb{R}^d$ is the initial node feature vector. Let $\mathbf{u}' \in \mathbb{R}^{d'}$ and $\mathbf{v} \in \mathbb{R}^{d'}$ be embeddings of nodes u' and v, accordingly. Let $s_{\mathbf{\Theta}}(u,v) = \log\sigma(\mathbf{u}^\top\mathbf{v})$ and $\tilde{s}_{\mathbf{\Theta}}(u',v) = \log(1-\sigma(\mathbf{u}'^\top\mathbf{v}))$, where $\sigma(\cdot)$ is the sigmoid function. Then, one obtains the contrastive objective to maximize employed by LINE [29], GraphSAGE [10] and many other methods according to the analysis of Yang *et al*. [35]:

$$J(\mathbf{\Theta}) = \mathbb{E}_{v\sim p_d(v)}\left[\mathbb{E}_{u\sim p_d(u|v)} \log\sigma(\mathbf{u}^\top\mathbf{v}) + \eta \mathbb{E}_{u'\sim p_n(u'|v)} \log\sigma(-\mathbf{u}'^\top\mathbf{v})\right]. \tag{3}$$

In what follows, we argue that the choice of sigmoid for $\sigma(\cdot)$ has negative consequences, and we derive COLES under a different choice of $s_\Theta(v,u)$ and $\tilde{s}_\Theta(v,u')$.

3 Methodology

Below we depart from the typical contrastive sampling and we derive our COntrastive Laplacian EigenmapS (COLES).

3.1 Contrastive Laplacian Eigenmaps

Instead of log-sigmoid used in $s_\Theta(v,u)$ and $\tilde{s}_\Theta(v,u')$ of Eq. (3), let us substitute $s_\Theta(v,u) = \log\exp(\mathbf{u}^\top\mathbf{v}) = \mathbf{u}^\top\mathbf{v}$ and $\tilde{s}_\Theta(v,u') = \log\exp(-\mathbf{u}'^\top\mathbf{v}) = -\mathbf{u}'^\top\mathbf{v}$ into Eq. (2), which yields:

$$J(\mathbf{\Theta}) = \mathbb{E}_{v\sim p_d(v)}\left[\mathbb{E}_{u\sim p_d(u|v)}(\mathbf{u}^\top\mathbf{v}) + \eta \mathbb{E}_{u'\sim p_n(u'|v)}\left(-\mathbf{u}'^\top\mathbf{v}\right)\right]. \tag{4}$$

We assume that variables of the above sampling objective to maximize can be constrained (*e.g.*, the ℓ_2 norms to prevent ill-posed solutions) and represented by degree-normalized adjacency matrices. Thus, we can cast Eq. (4) into the objective of COLES (kindly see our Suppl. Material for derivations):

$$\begin{aligned}\mathbf{Y}^* &= \underset{\mathbf{Y},\text{ s.t. }\mathbf{Y}^\top\mathbf{Y}=\mathbf{I}}{\arg\min}\ \mathrm{Tr}(\mathbf{Y}^\top\mathbf{L}\mathbf{Y}) - \frac{\eta'}{\kappa}\sum_{k=1}^{\kappa}\mathrm{Tr}(\mathbf{Y}^\top\mathbf{L}_k^{(-)}\mathbf{Y}) \\ &= \underset{\mathbf{Y},\text{ s.t. }\mathbf{Y}^\top\mathbf{Y}=\mathbf{I}}{\arg\max}\ \mathrm{Tr}(\mathbf{Y}^\top\Delta\mathbf{W}\mathbf{Y}) \quad \text{where} \quad \Delta\mathbf{W} = \mathbf{W}^{(+)} - \frac{\eta'}{\kappa}\sum_{k=1}^{\kappa}\mathbf{W}_k^{(-)},\end{aligned} \tag{5}$$

matrix $\mathbf{Y} \in \mathbb{R}^{n\times d'}$ contains row-wise embedding vectors, $\mathbf{L}_k^{(-)}$ for $k = 1,\cdots,\kappa$ are randomly generated degree-normalized Laplacian matrices capturing the negative sampling, $\mathbf{L}_k^{(-)} = \mathbf{I} - \mathbf{W}_k^{(-)}$ and $\mathbf{L} = \mathbf{I} - \mathbf{W}^{(+)}$. Moreover, $0 \leq \eta' \leq 1$ controls the impact of $\mathbf{W}_k^{(-)}$ and enables for the solution to exist by ensuring $\Delta\mathbf{W} \in \mathbb{S}_+^n$. Another common approach is to truncate the negative spectrum.

We note that COLES minimizes over the standard Laplacian Eigenmap while maximizing over the randomized Laplacian Eigenmap, which alleviates the lack of negative sampling in the original Laplacian Eigenmaps. However, unlike Laplacian Eigenmaps, we do not optimize over free variables $\mathbf{Y}$ but over the network parameters, as in Eq. (1) and (6). It is easy to note that for $\eta' = 0$, Eq. (5) reduces to standard Laplacian Eigenmaps [3] given by $\mathbf{Y}^* = \arg\min_{\mathbf{Y}} \mathrm{Tr}(\mathbf{Y}^\top\mathbf{L}\mathbf{Y})$, s.t. $\mathbf{Y}^\top\mathbf{Y} = \mathbf{I}$.

COLES for Linear Graph Networks. In what follows, we are especially interested in the lightweight family of LGNs such as SGC [34] and S^{2}GC [40] (APPNP [17] with the linear activation could be another choice), whose COLES-based objective can be reformulated as:

$$\mathbf{P}^* = \underset{\mathbf{P},\text{ s.t. }\mathbf{P}\mathbf{P}^\top=\mathbf{I}}{\arg\max}\ \mathrm{Tr}(\mathbf{P}\mathbf{X}^\top\mathbf{F}^\top\Delta\mathbf{W}\mathbf{F}\mathbf{X}\mathbf{P}^\top), \tag{6}$$

where $\mathbf{F} \in \mathbb{S}_+^n$ is the so-called spectral filter operating on the degree-normalized graph adjacency matrix, whereas $\mathbf{P} \in \mathbb{R}^{d' \times d}$ is a unitary projection matrix such that $0 < d' < d$. The solution to Eq. (6) can be readily obtained by solving the generalized eigenvalue problem $\mathbf{X}^\top \mathbf{F}^\top \Delta \mathbf{W} \mathbf{F} \mathbf{X} \mathbf{p} = \lambda \mathbf{p}$ (an SVD on a small $d \times d$ matrix $(\mathbf{X}^\top \mathbf{F}^\top \Delta \mathbf{W} \mathbf{F} \mathbf{X}) \in \mathbb{S}_+^d$). As a result, we obtain a matrix of embeddings $f_{\mathbf{P}}(\mathbf{X}) = \mathbf{F}\mathbf{X}\mathbf{P}^\top \in \mathbb{R}^{n \times d'}$. The spectral filter $\mathbf{F}$ for SGC and S^{2}GC given a degree-normalized graph adjacency matrix $\mathbf{W} \in \mathbb{S}_+^n$ is given as $\mathbf{W}^K$ and $\alpha \mathbf{I} + \frac{1-\alpha}{K} \sum_{k=1}^{K} \mathbf{W}^k$, respectively. Here, integer $K \geq 1$ and scalar $\alpha \geq 0$ are the number of layers and the importance of self-loop. It is easy to notice that Eq. (6) is related to Locality Preserving Projections [12] if $\eta' = 0$. Finally, we note that enforcing the orthogonality constraints in Eq. (6) coincides with the SVD-based solution described above, whereas in the most general form of COLES in Eq. (1), one may have to resort to the regularization or constraints imposed on minibatches (depending on the backbone).

4 Theoretical Analysis

4.1 COLES is Wasserstein-based Contrastive Learning

By casting the positive and negative distributions of SampledNCE as the real and generated data distributions of GAN, the key idea of this analysis is to (i) cast the traditional contrastive loss in the Eq. (3) (used by LINE [29], GraphSAGE [10] and other methods [35]) as a GAN framework, and show this corresponds to the use of JS divergence and (ii) cast the objective of COLES in the Eq. (4) as a GAN framework, and show it corresponds to the use of a surrogate of Wasserstein distance. The latter outcome is preferable as under the poor overlap of two distributions, the JS div. yields zero leading to vanishing gradients, and the Wasserstein distance is known not to suffer from this issue.

For simplicity of derivations, let $\eta = 1$ and recall that components $\log(\sigma(-\mathbf{u}'^\top \mathbf{v})) = \log(1 - \sigma(\mathbf{u}'^\top \mathbf{v}))$ in Eq. (3), where σ is the sigmoid function. Now, let 'discriminator' $D(x) = \sigma(x)$ where $x = \mathbf{u}^\top \mathbf{v}$ if x is sampled from the 'real' distribution $p_r = p_d(u \mid v)$ and $x = \mathbf{u}'^\top \mathbf{v}$ if x is sampled from the 'generator' distribution $p_g = p_n(u' \mid v)$. Following considerations of [33], one arrives at a GAN-inspired formulation which depends on the choice of 'discriminator' $D(x)$:

$$J^*(D) = \int_x \Big(p_r(x) \log(D(x)) + p_g(x) \log(1 - D(x)) \Big) \mathrm{d}x. \tag{7}$$

Looking for the optimal function $D(x)$ that maximizes Eq. (7), [33] arrived at solution:

$$D^*(x) = \frac{p_r(x)}{p_r(x) + p_g(x)} \in [0, 1], \tag{8}$$

For p_g getting very close to p_r, which would be the case when our negative sampling yields hard negatives, optimal $D^*(x)$ becomes $\frac{1}{2}$ which yields the solution:

$$J^*(D^*) = 2JS\left(p_r \| p_g\right) - 2\log 2, \tag{9}$$

where $JS\left(p_r \| p_g\right) = \frac{1}{2} KL\left(p_r \| \frac{p_r + p_g}{2}\right) + \frac{1}{2} KL\left(p_g \| \frac{p_r + p_g}{2}\right)$. Thus, for the optimal $D^*(x)$, one can think of traditional contrastive losses to enjoy the JS divergence. However, if p_r and p_g are even locally disjoint (*e.g.*, positive and negative samples in the minibatch of the SGD optimization), Eq. (9) becomes a constant that does not contribute to learning of parameters of network, as the JS divergence yields zero for disjoint distributions. Figure 1 shows densities of $x = \mathbf{u}^\top \mathbf{v}$ and $x = \mathbf{u}'^\top \mathbf{v}$ for p_r and p_d estimated by the Parzen window on two sampled minibatches of contrastive GCN. As is clear, these distributions are approximately disjoint.

Compared with the JS divergence, the Wasserstein distance changes smoothly even between disjoint distributions:

$$\mathrm{Wa}(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(\mathbf{x}, \mathbf{x}') \sim \gamma} \|\mathbf{x} - \mathbf{x}'\|_1, \tag{10}$$

where $\Pi(p_r, p_g)$ is the set of all possible joint distributions coupling p_r and p_g, and each $\gamma \in \Pi(p_r, p_g)$ forms one transportation plan. As it is intractable to exhaust all possible transportation plans $\Pi(p_r, p_g)$, a popular solution is to employ the Kantorovich-Rubinstein duality, formulated for COLES as:

$$\begin{aligned} \mathrm{Wa}(p_r, p_g) &= \frac{1}{K'} \max_{\boldsymbol{\Theta}} \left(\mathbb{E}_{\mathbf{x} \sim p_r}[g_{\boldsymbol{\Theta}}(\mathbf{x})] - \mathbb{E}_{\mathbf{x}' \sim p_g}[g_{\boldsymbol{\Theta}}(\mathbf{x}')] \right) \\ &\geq \max_{\boldsymbol{\Theta}} \left[\mathbb{E}_{u \sim p_d(u|v)} (\mathbf{u}^\top \mathbf{v}) + \mathbb{E}_{u' \sim p_n(u'|v)} (-\mathbf{u}'^\top \mathbf{v}) \right], \end{aligned} \tag{11}$$

under a drawn anchor $v \sim p_d(v)$, where Wa is the Wasserstein distance, and $g(\cdot)$ must be a Lipschitz continuous function, that is, $\|g(\mathbf{x}) - g(\mathbf{x}')\|_1 \leq K'\|\mathbf{x} - \mathbf{x}'\|_1$ for this duality to hold. In our case, after the swap of distributions p_r and p_g for p_d and p_n, we obtain the inequality because that we choose a specific g given by $g_v(\mathbf{u}) = \mathbf{u}^\top \mathbf{v}$. Note that $\mathbf{v}$ is parametrized by the neural network with parameters $\boldsymbol{\Theta}$, thus it can enumerate a subset of functions, which satisfies the Lipschitz constant K'.

Lipschitz continuity of COLES. In order to assure the Lipschitz continuity of COLES, let individual embeddings be stacked row-wise into a matrix and ℓ_2-norm normalized along rows, or along columns. Given v (the reference node), the following holds:

$$|\mathbf{u}^\top \mathbf{v} - \mathbf{u}'^\top \mathbf{v}| \leq \|\mathbf{v}\|_{\max} \|\mathbf{u} - \mathbf{u}'\|_1 \leq K'\sqrt{d'}\|\mathbf{u} - \mathbf{u}'\|_2,$$

where $K' = \max_v \|\mathbf{v}\|_{\max}$ (≤ 1 in the case of sphere embedding as well as the constraint $\mathbf{Y}^\top \mathbf{Y} = \mathbf{I}$ of the COLES formulation in Eq. (5)). Thus, the function $g(\mathbf{u}) = \mathbf{u}^\top \mathbf{v}$ is Lipschitz with constant $K'\sqrt{d'}$.

4.2 COLES enjoys the Block-contrastive Loss

We notice that COLES leverages an access to blocks of similar data, rather than just individual pairs in the loss function. To this end, we resort to the Prop. 6.2 of Saunshi *et al*. [27], which shows that for family of functions $\mathcal{F}$ whose $\|f(\cdot)\| \leq R$ for some $R > 0$, a block-contrastive loss L_{un}^{block} is bounded by a pairwise-contrastive loss L_{un}, that is, $L_{un}^{\text{block}}(f) \leq L_{\text{un}}(f)$, thus attaining a better minimum.

To this end, we show that COLES is a block-contrastive loss, which explains a good performance of COLES. Following Eq. (4), for a given embedding $\mathbf{v} = f_{\boldsymbol{\Theta}}(\mathbf{x}_v)$, and b embeddings $\mathbf{u}_i = f_{\boldsymbol{\Theta}}(\mathbf{x}_{u_i})$ and $\mathbf{u}'_i = f_{\boldsymbol{\Theta}}(\mathbf{x}_{u'_i})$ drawn according to $p_d(u \mid v)$ and $p_n(u' \mid v)$, we have (note minus preceding eq. as here we minimize):

$$-\mathbb{E}_{u \sim p_d(u|v)}(\mathbf{u}^\top \mathbf{v}) + \mathbb{E}_{u' \sim p_n(u'|v)}\left(-\mathbf{u}'^\top \mathbf{v}\right) = -\mathbf{v}^\top \left(\frac{\sum_i \mathbf{u}_i}{b} - \frac{\sum_i \mathbf{u}'_i}{b}\right) = -\mathbf{v}^\top(\boldsymbol{\mu}^+ - \boldsymbol{\mu}^-), \quad (12)$$

where $\boldsymbol{\mu}^+$ and $\boldsymbol{\mu}^-$) are positive and negative block summaries of sampled nodes. Looking at Eq. (5), it is straightforward to simply expand $\sum_{(i,j)\in E} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 \, \Delta W_{ij}$ to see that each index i will act as a selector of anchors, whereas index j will loop over positive and negative samples taking into account their connectivity to i captured by ΔW_{ij}. We leave this expansion out for the Suppl. Material.

5 Related Works

Graph Embeddings. The motivation for early graph embedding algorithms such as Laplacian Eigenmaps [3] and IsoMap [30] was to reduce the high dimensionality of representations by assuming the data lies on a low-dimensional manifold. With these techniques, given a set of high-dimensional data features, one can construct a similarity graph based on the pairwise feature similarity, and embed each node into a low-dimensional space. The input is a graph constructed from non-relational high dimensional data features. For example, Laplacian Eigenmaps ignore the relation between dissimilar node pairs, that is, dissimilar nodes are not penalized for their dissimilarity.

To alleviate the above shortcomings, DeepWalk [24] uses truncated random walks to explore the network structure and utilizes the skip-gram model [21] for word embedding to derive the embedding vectors of nodes. LINE [29] explores a similar idea with an explicit objective function by setting the walk length as one and applying negative sampling [21]. Finally, Node2Vec [9] interpolates between breadth- and depth-first sampling strategies to aggregate different types of neighborhoods.

Representation Learning for Graph Neural Networks. Recent works on GNNs employ multiple layers of graph convolutions. Among them, considerable literature has grown around the theme of (semi-)supervised GNN [16], which requires labelled datasets that may not be accessible in real-world applications. In contrast, unsupervised GNNs have received less attention. GCN [16] employs the minimization of reconstruction error as the objective function to train the encoder. GraphSAGE [10] incorporates objectives inspired by DeepWalk *e.g.*, contrastive loss encouraging nearby nodes to have similar representations while preserving dissimilarity between representations of disparate nodes. Compared with previous objective functions, DGI [32], inspired by Deep InfoMax (DIM) [13],

proposes an objective with global-local sampling strategy, which maximizes the Mutual Information (MI) between global and local graph embeddings. In contrast, Augmented Multiscale Deep InfoMax (AMDIM) [2] maximizes MI between multiple views of data. MVRLG [11] contrasts encodings from first-order neighbors and a graph diffusion. MVRLG uses GCNs to learn node embeddings for different views. We note that the multi-view and data augmentation approaches are not in the scope of our paper (they are complementary to COLES). We note also a family of linear networks *e.g.*, SGC [34] and S^{2}GC [40] capture a neighborhood and increasingly larger neighborhoods of each node, respectively. Compared with dedicated node embedding methods, SGC and S^{2}GC have no projection layer, which results in embeddings of size equal to the input dimension. Finally, we note that DGI [32] uses the block-contrastive strategy [27] but it treats negative samples as a difference of instances and it uses a summary of node embeddings for positive samples.

(Negative) Sampling. Sampling node pairs may rely on random walks [24] or second-order proximity [29], *etc*. In contrast, COLES samples an undirected graph based on the random graph sampling theory [8], where each edge is independently chosen with the probability $p' > 0$.

6 Experiments

Below, we evaluate the performance of COLES on both transductive and inductive node classification tasks. Node clustering is also considered in evaluations. COLES is compared with state-of-the-art unsupervised, contrastive and (semi-)supervised methods, where unsupervised methods do not use label information except for the classifier, contrastive approaches use the contrastive setting to learn similarity/dissimilarity, while (semi-)supervised methods use labels to train their projection layers and a classifier. By 'semi-supervised', we mean that not all nodes are labelled at the training time even though their features are used during training. (Semi-)supervised models use a softmax classifier, whereas unsupervised and contrastive methods use a logistic regression classifier.

Datasets. COLES is evaluated on four citation networks: Cora, Citeseer, Pubmed, Cora Full [16, 4] for transductive setting. We also employ ogbn-arxiv from open graph benchmark OGB [14] to illustrate the potential of COLES on large-scale graph-structured data. Finally, the community networks Reddit [37] is used in inductive setting. Details of all datasets are given in Table 1.

Metrics. Fixed data splits [36] for transductive tasks are often used in evaluations between different models. However, such experimental setup may benefit models that easily overfit [28]. Thus, instead of fixed data splits, results are averaged over 50 random splits for each dataset and standard deviations are reported for empirical evaluation on transductive tasks. Moreover, we also test the performance under different number of samples per class *i.e.*, 5 and 20 samples each class. Typically, the performance for the inductive task is tested on relatively larger graphs. Thus, we choose fixed data splits as in previous papers [10, 37], and we report the Micro-F1 scores averaged on 10 runs.

Baseline models. We group baseline models into unsupervised, contrastive and (semi-)supervised methods, and implement them in the same framework/testbed. Contrastive methods include DeepWalk [24], GCN+SampledNCE, developed in analogy to GraphSAGE+SampledNCE [10], Graph2Gauss [4], DGI [32] and GRACE [41], which are the main competitors. Please note that GRACE is based on multi-view and data augmentation. We also compare results with some representative (semi-)supervised models, namely GCN [16], GAT [32] and MixHop [1]. In contrast, SGC and S^{2}GC are examples of unsupervised spectral filters acting on features. They do not have any learnable parameters that depend on labels, with exception of a classifier. To reduce the resulting dimensionality, we also add PCA-S^{2}GC and RP-S^{2}GC, which use PCA and random projections to obtain the projection layer on these methods. We extend our COLES framework with different GNNs: GCN, SGC and/or S^{2}GC, and we name them COLES-GCN, COLES-SGC and COLES-S^{2}GC. As COLES-GCN is a multi-layer non-linear encoder, the optimization of COLES-GCN is non-convex. The optimization of COLES-SGC and COLES-S^{2}GC is convex if $\Delta\mathbf{W} \in \mathbb{S}_+^n$. We follow the common practice from the original papers regarding setting hyperparameters.

General model setup. For all (semi-)supervised models, we use early stopping on each random split and we capture the corresponding classification result. For all unsupervised models, we choose the embedding dimension to be 512 on Cora, Citeseer and Cora Full, and 256 on Pubmed. After the embeddings of nodes are learnt, a classifier is trained by applying the logistic regression in the embedding space. For inductive learning, methods based on COLES use 512-dimensional embeddings. Other hyperparameters for the baseline models are the same as in original papers.

Table 1: The statistics of datasets.

Dataset	Task	Nodes	Edges	Features	Classes
Cora	Transductive	2,708	5,429	1,433	7
Citeseer	Transductive	3,327	4,732	3,703	6
Pubmed	Transductive	19,717	44,338	500	3
Cora Full	Transductive	19,793	65,311	8,710	70
OGB-Arxiv	Transductive	169,343	1,166,243	128	40
Reddit	Inductive	232,965	11,606,919	602	41

Hyperparameter of our models. In the transductive experiments, the detailed hyperparameter settings for Cora, Citeseer, Pubmed, and Cora Full are listed below. For COLES framework, we use the Adam optimizer with learning rates of $[0.001, 0.0001, 0.02, 0.02]$ and the decay of $[5e-4, 1e-3, 5e-4, 2e-4]$. The number of training epochs are $[20, 20, 100, 30]$, respectively. We sample 10 randomized adjacent matrices, 5 negative samples for each node in each matrix on each dataset before training. For the S^{2}GC and COLES-S^{2}GC, the number of propagation steps (layers) are 8 for all datasets except Cora Full (2 steps). For SGC and COLES-SGC, we use 2 steps for all datasets.

Table 2: Mean classification accuracy (%) and the standard dev. over 50 random splits. Numbers of labeled samples per class are in parentheses. The best accuracy per column is in bold. Models are grouped into semi-supervised, contrastive and unsupervised. OOM means out of memory.

Method		Cora		Citeseer		Pubmed		Cora Full	
		(5)	(20)	(5)	(20)	(5)	(20)	(5)	(20)
Semi-supervised	GCN	67.5±4.8	79.4±1.6	57.7±4.7	69.4±1.4	65.4±5.2	77.2±2.1	49.3±1.8	61.5±0.5
	GAT	71.2±3.5	79.6±1.5	54.9±5.0	69.1±1.5	65.5±4.6	75.4±2.3	43.9±1.5	56.9±0.6
	MixHop	67.9±5.7	80.0±1.4	54.5±4.3	67.1±2.0	64.4±5.6	75.7±2.7	47.5±1.5	61.0±0.7
Contrastive	DeepWalk	60.3±4.0	70.5±1.9	38.3±2.9	45.6±2.0	60.3±5.6	70.8±2.6	38.9±1.4	51.1±0.7
	GCN+SampledNCE	61.3±4.3	74.3±1.6	42.3±3.4	56.8±1.9	60.9±5.7	70.3±2.5	32.7±1.9	45.2±0.9
	SAGE+SampledNCE	65.0±3.5	73.8±1.5	48.0±3.5	56.5±1.6	64.1±6.1	74.6±1.9	35.0±1.4	43.6±0.6
	Graph2Gauss	72.7±2.0	76.2±1.1	60.7±3.5	65.7±1.5	67.6±3.9	74.1±2.1	38.9±1.3	49.3±0.5
	DGI	72.9±4.0	78.1±1.8	65.7±3.6	71.1±1.1	65.3±5.7	73.9±2.3	50.5±1.4	58.4±0.6
	GRACE	64.9±4.2	73.9±1.6	61.8±3.9	68.4±1.6	OOM	OOM	OOM	OOM
	COLES-GCN	73.8±3.4	80.8±1.3	66.0±2.6	69.0±1.3	62.7±4.6	72.7±2.1	47.3±1.5	58.9±0.5
	COLES-S^{2}GC	**76.5±2.6**	**81.5±1.2**	**67.5±2.2**	**71.3±1.0**	66.0±5.2	**77.4±1.9**	**50.8±1.4**	**61.8±0.5**
Unsupervised	SGC	63.9±5.4	78.3±1.9	59.5±3.4	69.8±1.4	65.8±4.4	76.3±2.3	46.0±2.2	57.7±1.2
	S^{2}GC	71.4±4.4	81.3±1.2	60.3±4.0	69.5±1.2	**67.6±4.2**	73.3±2.0	41.8±1.7	60.0±0.5
	PCA-S^{2}GC	72.1±3.8	81.2±1.3	61.0±3.5	68.8±1.3	67.5±4.3	73.2±2.0	42.3±1.7	59.3±0.6
	RP-S^{2}GC	65.9±4.6	78.1±1.2	51.4±3.2	61.7±1.6	66.1±5.0	72.5±1.9	31.5±1.4	48.7±0.6

Transductive Learning In this section, we consider transductive learning where all nodes are available in the training process.

Contrastive Embedding Baselines *vs*. COLES. Table 2 shows that the performance of COLES-GCN and the linear variant, COLES-S^{2}GC, are better than other unsupervised models. In particular, COLES-GCN outperforms GCN+SampledNCE on all four datasets, which shows that COLES has an advantage over the SampledNCE framework. In addition, COLES-S^{2}GC typically outperforms the best contrastive baseline DGI by up to 3.4%. In Cora full, we notice that S^{2}GC baseline underperforms when training with 5 samples. However, COLES-S^{2}GC is able to significantly boost its performance by 9%. On Citeseer with 5 training samples, COLES-S^{2}GC outperforms S^{2}GC by 6.8%. Finally, Table 6 shows that COLES-S^{2}GC outperforms all other methods on the challenging ogbn-arxiv, while using a very small number of trainable parameters.

Semi-supervised GNNs *vs*. COLES. Table 2 shows that the traditional contrastive GCN baselines perform worse than semi-supervised variants, especially when 20 labelled samples per class are available. In contrast, COLES-GCN outperformed the semi-supervised GCN on Cora by 6.3% and 1.4% given 5 and 20 labelled samples per class. COLES-GCN also outperformed GCN on Citeseer and Cora Full by 8.3% and 6.3% given 5 labelled samples per class. When the number of labels per class is 5, COLES-S^{2}GC outperforms GCN by a margin of 8.1% on Cora and 9.4% on Citeseer. These results clearly demonstrate the advantage of COLES on four datasets when the number of samples per class is 5. Even for 20 labeled samples per class, COLES-S^{2}GC outperforms the best semi-supervised baselines on all four datasets *e.g.*, by 1.7% on Citeseer. Semi-supervised models are greatly affected by the low number of labeled samples available, which is also observed in [20], especially for GAT and MixHop models. The accuracy of COLES-GCN and COLES-S^{2}GC are not affected as significantly due to the contrastive setting.

Table 3: The influence of the number (κ) of negative Laplacian graphs on COLES-S^{2}GC. Parentheses show the no. of labeled samples p/class.

		κ 0	1	5	10
Cora	(20)	79.88	81.43	81.18	81.17
Cora	(5)	70.12	76.24	75.89	75.79
Citeseer	(20)	69.42	70.71	70.61	70.61
Citeseer	(5)	58.17	67.03	66.96	67.04

Table 4: The influence of the number (κ) of negative Laplacian graphs on COLES-GCN. Parentheses show the no. of labeled samples p/class.

		κ 0	1	5	10
Cora	(20)	75.70	80.90	80.87	80.90
Cora	(5)	60.97	74.14	74.11	74.07
Citeseer	(20)	60.61	69.04	69.21	69.08
Citeseer	(5)	45.31	65.85	66.08	66.01

Table 5: Test Micro F1 Score (%) averaged over 10 runs on Reddit. Results of other models are from original papers.

Setting	Model	Test F1
Supervised	SAGE-mean	95.0
	SAGE-LSTM	**95.4**
	SAGE-GCN	93.0
Contrastive	SAGE-mean	89.7
	SAGE-LSTM	90.7
	SAGE-GCN	90.8
	FastGCN	93.7
	DGI	94.0
	COLES-GCN	94.0
	COLES-SGC	94.8
	COLES-S^{2}GC	**95.4**

Table 6: Mean classification accuracy (%) and the standard dev. over 10 runs on ogbn-arxiv. Results of other models are from original papers.

Method		Test Acc.	#Params
MLP		55.50±0.23	110,120
Node2Vec	[9]	70.07±0.13	21,818,792
GraphZoom	[7]	71.18±0.18	8,963,624
C&S	[15]	71.26±0.01	5,160
SAGE-mean	[10]	71.49±0.27	218,664
GCN	[16]	71.74±0.29	142,888
DeeperGCN	[19]	71.92±0.17	491,176
SIGN	[26]	71.95±0.11	3,566,128
FrameLet	[39]	71.97±0.12	1,633,183
S^{2}GC	[40]	72.01±0.25	110,120
COLES-S^{2}GC		**72.48±0.25**	110,120

Unsupervised GNNs *vs*. COLES. SGC and S^{2}GC are unsupervised LGNs as they are spectral filters which do not use labels (except for the classifier). Table 2 shows that COLES-S^{2}GC outperforms RP-S^{2}GC and PCA-S^{2}GC under the same size of projections. In most cases, COLES-S^{2}GC also outperforms the unsupervised S^{2}GC baseline (high-dimensional representation).

Negative Laplacian Eigenmaps. Below, we analyze how κ in Eq. (5) influences the performance. We set $\kappa \in \{0, 1, 5, 10\}$ on COLES-S^{2}GC and COLES-GCN given Cora and Citeseer with 5 and 20 labeled samples per class. The case of $\kappa = 0$ means no negative Laplacian Eigenmaps are used and the solution simplifies to regular Laplacian Eigenmaps parametrized by GCN embeddings. Table 3 shows that without the negative Laplacian Eigenmaps, the COLES-S^{2}GC performance drops significantly *i.e.*, between 6% and 9% for 5 labeled samples per class. That means the negative Laplacian Eigenmaps play important role which highlights the benefit of COLES. Although negative Laplacian Eigenmaps improve results, their number $\kappa > 1$ changes the performance little. Table 4 shows that COLES-GCN relies on negative Laplacian Eigenmaps. Without negative Laplacian Eigenmap, the performance of COLES-GCN drops by 20% on Citeseer with 5 samples per class. Even when 20 samples per class are used, if $\kappa = 0$, COLES-GCN looses up to 8.4% in performance.

6.1 Inductive Learning

In inductive learning, models have no access to the test set, thus they need to generalize well to unseen samples. Table 5 shows that COLES enjoys a significant performance gain (1% - 5% in Micro-F1 scores), performing close to supervised methods and enjoying a low memory footprint. In contrast, DGI on Reddit did not fit in our Nvidia GTX 1080 GPU (94.0 Micro-F1 is taken from [32]).

Scalability. Methods such as GraphSAGE and DGI use GCNs as their building blocks, and in order to scale to a large graph, they need neighbor sampling strategies (limited GPU memory). Under such a trade-off strategy, several redundant forward and backward propagations are conducted on each node during each epoch, which results in a long training time. However, LGNs combined with COLES, that is, COLES-SGC and COLES-S^{2}GC) enjoy a straightforward implementation which reduces the memory usage and training time significantly. For graphs with more than 100 thousands nodes and 10 millions edges (Reddit), our model runs smoothly on a simple NVIDIA 1080 GPU. For even larger graph datasets, the closed-form solution is competitive because for COLES-SGC

Table 7: The clustering performance on Cora, Citeseer and Pubmed.

Method	Input	Cora			Citeseer			Pubmed		
		Acc%	NMI%	F1%	Acc%	NMI%	F1%	Acc%	NMI%	F1%
k-means	Feature	34.65	16.73	25.42	38.49	17.02	30.47	57.32	29.12	57.35
Spectral-f	Feature	36.26	15.09	25.64	46.23	21.19	33.70	59.91	32.55	58.61
Spectral-g	Graph	34.19	19.49	30.17	25.91	11.84	29.48	39.74	3.46	51.97
DeepWalk	Graph	46.74	31.75	38.06	36.15	9.66	26.70	61.86	16.71	47.06
GAE	Both	53.25	40.69	41.97	41.26	18.34	29.13	64.08	22.97	49.26
VGAE	Both	55.95	38.45	41.50	44.38	22.71	31.88	65.48	25.09	50.95
ARGE	Both	64.00	44.90	61.90	57.30	35.00	54.60	59.12	23.17	58.41
ARVGE	Both	62.66	45.28	62.15	54.40	26.10	52.90	58.22	20.62	23.04
GCN	Both	59.05	43.06	59.38	45.97	20.08	45.57	61.88	25.48	60.70
SGC	Both	62.87	50.05	58.60	52.77	32.90	63.90	69.09	31.64	68.45
S^2GC	Both	68.96	54.22	**65.43**	69.11	42.87	64.65	68.18	31.82	67.81
COLES-GCN	Both	60.74	45.49	59.33	63.28	37.54	59.17	63.46	25.73	63.42
COLES-SGC	Both	65.62	52.32	56.95	68.24	43.09	63.85	**69.47**	32.31	**68.57**
COLES-S^2GC	Both	**69.70**	**55.35**	63.06	**69.20**	**44.41**	**64.70**	68.76	**33.42**	68.12

and COLES-S^2GC, the cost of eigen-decomposition depends on the target and the feature input dimensions. For Reddit, only few seconds are spent on eigen-decomposition.

6.2 Node Clustering

We compare COLES-GCN and COLES-S^2GC with three types of clustering methods listed below:

i. Methods that use only node features *e.g.*, k-means and spectral clustering (spectral-f) that constructs a similarity matrix with the node features by a linear kernel.

ii. Structural clustering methods that use only the structure of graph: spectral clustering (spectral-g) that takes the node adjacency matrix as the similarity matrix, and DeepWalk [24].

iii. Attributed graph clustering methods that use node features and the graph: Graph Autoencoder (GAE), Graph Variational Autoencoder (VGAE) [16], Adversarially Regularized Graph Autoencoder (ARGE), Variational Graph Autoencoder (ARVGE) [23], SGC [34] and S^2GC [40].

We measure the performance by the clustering Accuracy (Acc), Normalized Mutual Information (NMI) and macro F1-score (F1). We run each method 10 times on Cora, CiteSeer and PubMed. We report the clustering results in Table 7. We set the number of propagation steps to 8 for SGC, S^2GC, COLES-SGC and COLES-S^2GC, following the setting of [38]. We note that COLES-S^2GC outperforms S^2GC in most cases, whereas COLES-GCN outperforms contrastive GCN on all three datasets.

7 Conclusions

We have proposed a new network embedding called COnstrative Laplacian EigenmapS (COLES), which recognizes the importance of negative sample pairs in Laplacian Eignemaps, and combined with GCN, SGC and S^2GC backbones, outperforms many unsupervised, contrastive and (semi-)supervised methods. Our contrastive parametric graph embeddings works with numerous backbones. We shown that the SampledNCE framework with the sigmoid non-linearity yields the JS divergence during the analysis with a GAN-inspired reformulation. However, COLES uses the RBF non-linearity, which results in the Kantorovich-Rubinstein duality; COLES essentially minimizes a surrogate of Wasserstein distance, which offers a reasonable transportation plan, and helps avoid pitfalls of the JS divergence. In contrast, SampledNCE relies on the JS divergence, thus suffering when the positive and negative distributions scarcely overlap. Finally, we have shown that COLES enjoys the so-called block-contrastive loss known for a better performance than the family of pair-wise contrastive losses.

References

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning*, pages 21–29, 2019.

[2] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019.

[3] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[4] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

[5] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[6] Miguel Á. Carreira-Perpiñan. The elastic embedding algorithm for dimensionality reduction. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 167–174, 2010.

[7] C Deng, Z Zhao, Y Wang, Z Zhang, and Z Feng. Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding. In *The International Conference on Learning Representations (ICLR)*, 2020.

[8] Paul ErdHos and Alfréd Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[9] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

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

[12] Xiaofei He and Partha Niyogi. Locality preserving projections. *Advances in neural information processing systems*, 16(16):153–160, 2004.

[13] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[14] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[15] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin Benson. Combining label propagation and simple models out-performs graph neural networks. In *International Conference on Learning Representations*, 2021.

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

[17] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019.

[18] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, 27:2177–2185, 2014.

[19] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

[20] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 3538–3545. Association for the Advancement of Artificial Intelligence, 2018.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[23] S Pan, R Hu, G Long, J Jiang, L Yao, and C Zhang. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[24] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[25] Ignacio Ramirez, Pablo Sprechmann, and Guillermo Sapiro. Classification and clustering via dictionary learning with structured incoherence and shared features. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3501–3508, 2010.

[26] Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020.

[27] Nikunj Saunshi, Orestis Plevrakis, Sanjeev Arora, Mikhail Khodak, and Hrishikesh Khandeparkar. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, pages 5628–5637. PMLR, 2019.

[28] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[29] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[30] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

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

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

[33] Lilian Weng. From gan to wgan. *arXiv preprint arXiv:1904.08994*, 2019.

[34] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

[35] Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. Understanding negative sampling in graph representation learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1666–1676, 2020.

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

[37] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

[38] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. Attributed graph clustering via adaptive graph convolution. *arXiv preprint arXiv:1906.01210*, 2019.

[39] Xuebin Zheng, Bingxin Zhou, Junbin Gao, Yu Guang Wang, Pietro Lio, Ming Li, and Guido Montúfar. How framelets enhance graph neural networks. 2021.

[40] Hao Zhu and Piotr Koniusz. Simple spectral graph convolution. In *International Conference on Learning Representations*, 2021.

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

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] Datasets are publicly available, with licenses included. We include the license to our code.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

i. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] W also add a paragraph on novelty in line 61.
   (b) Did you describe the limitations of your work? [Yes] See for instance lines 116–117 and 146–157.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We are not aware of any.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

ii. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Lines 94–98 explain how to obtain the solution to COLES. Lines 152–157 explain when COLES enjoys the Kantorovich-Rubinstein duality.
   (b) Did you include complete proofs of all theoretical results? [Yes] Derivations of Eq. (5) from Eq. (4) are in the Suppl. Material. Proof that Eq. (11) holds is given in lines 152–157. Lines 168–171 of Section 4.2 are in the Suppl. Material.

iii. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the Suppl. Material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See lines 208–253 and the Suppl. Material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See line 303 and the Suppl. Material.

iv. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [No] Authors provided licenses with their datasets.

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Licenses of datasets we use permit the usage by us. The same concerns the codes.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We use publicly available datasets used over many years by the public now. They have not been identified as offensive.

v. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]