

Symmetric Graph Convolutional Autoencoder for Unsupervised Graph Representation Learning

Jiwoong Park¹ Minsik Lee² Hyung Jin Chang³ Kyuewang Lee¹ Jin Young Choi¹

¹ASRI, Dept. of ECE., Seoul National University

²Div. of EE., Hanyang University

³School of Computer Science, University of Birmingham

{ptywoong, kyuewang, jychoi}@snu.ac.kr, mleepaper@hanyang.ac.kr, h.j.chang@bham.ac.uk

Abstract

We propose a symmetric graph convolutional autoencoder which produces a low-dimensional latent representation from a graph. In contrast to the existing graph autoencoders with asymmetric decoder parts, the proposed autoencoder has a newly designed decoder which builds a completely symmetric autoencoder form. For the reconstruction of node features, the decoder is designed based on Laplacian sharpening as the counterpart of Laplacian smoothing of the encoder, which allows utilizing the graph structure in the whole processes of the proposed autoencoder architecture. In order to prevent the numerical instability of the network caused by the Laplacian sharpening introduction, we further propose a new numerically stable form of the Laplacian sharpening by incorporating the signed graphs. In addition, a new cost function which finds a latent representation and a latent affinity matrix simultaneously is devised to boost the performance of image clustering tasks. The experimental results on clustering, link prediction and visualization tasks strongly support that the proposed model is stable and outperforms various state-of-the-art algorithms.

1. Introduction

A graph, which consists of a set of nodes and edges, is a powerful tool to seek the geometric structure of data. There are various applications using graphs in the machine learning and data mining fields such as node clustering [26], dimensionality reduction [1], social network analysis [15], chemical property prediction of a molecular graph [7], and image segmentation [30]. However, conventional methods for analyzing a graph have several problems such as low computational efficiency due to eigendecomposition or singular value decomposition, or only showing a shallow relationship between nodes.

In recent years, an emerging field called *geometric deep learning* [2], generalizes deep neural network models to

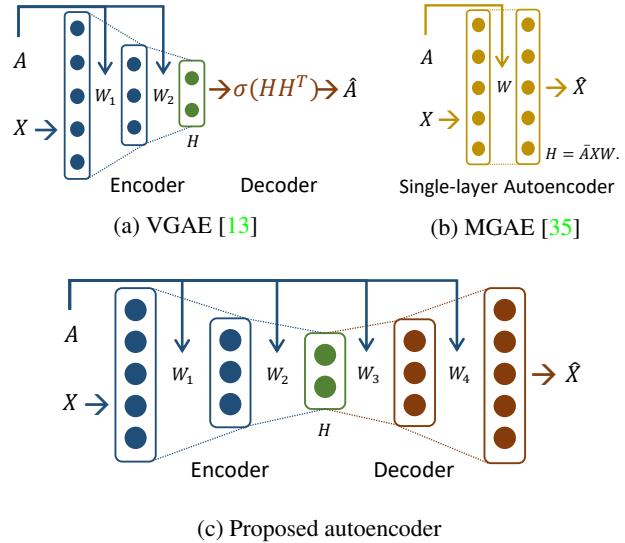


Figure 1: Architectures of existing graph convolutional autoencoders and proposed one. A , X , H and W denote the affinity matrix (structure of graph), node attributes, latent representations and the learnable weight of network respectively.

non-Euclidean domains such as meshes, manifolds, and graphs [14, 24, 20]. Among them, finding deep latent representations of geometrical structures of graphs using an autoencoder framework is getting growing attention. The first attempt is VGAE [13] which consists of a Graph Convolutional Network (GCN) [14] encoder and a matrix outer-product decoder as shown in Figure 1 (a). As a variant of VGAE, ARVGA [27] has been proposed by incorporating an adversarial approach to VGAE. However, VGAE and ARVGA were designed to reconstruct the affinity matrix A instead of node feature matrix X . Hence, the decoder part cannot be learnable, therefore, the graphical feature cannot be used at all in the decoder part. These facts can degrade the capability of graph learning. Following that, MGAE [35] has been proposed, which uses stacked single layer graph autoencoder with linear activation function and

marginalization process as shown in Figure 1 (b). However, since the MGAE reconstructs the feature matrix of nodes without hidden layers, it cannot manipulate the dimension of the latent representation and performs a linear mapping. This is a distinct limitation in finding a latent representation that clearly reveals the structure of the graph.

To overcome the limitation of the existing graph convolutional autoencoders, in this paper, we propose a novel graph convolutional autoencoder framework which has symmetric autoencoder architecture and uses both graph and node attributes in both the encoding and decoding processes as illustrated in Figure 1 (c). Our design of the decoder part is motivated from the analysis in a recent paper [19], that the encoder of VGAE [13] can be interpreted as a special form of Laplacian smoothing [32] that computes the new representation of each node as a weighted local average of neighbors and itself. This interpretation has inspired us to design a decoder to perform *Laplacian sharpening*, which is a counterpart of Laplacian smoothing. To realize a decoder to do Laplacian sharpening, we express Laplacian sharpening in the form of Chebyshev polynomial and newly reformulate it in a numerically stable form by utilizing a signed graph [18].

In computer vision fields, there is a popular assumption that, even though image datasets are high-dimensional in their ambient spaces, they usually reside in multiple low-dimensional subspaces [34]. Thus, especially for image clustering tasks, we apply the concept of subspace clustering, which has such an assumption about the input data in its own definition, to our graph convolutional autoencoder framework. Specifically, to find a latent representation and a latent affinity matrix simultaneously, we merge a subspace clustering cost function into the reconstruction cost of the autoencoder. Contrary to the conventional subspace clustering cost function [11, 39], we could derive a computationally efficient cost function.

The main contributions of this paper are summarized as follows:

- We propose the first completely symmetric graph convolutional autoencoder which utilizes both the structure of the graph and node attributes through the whole encoding-decoding process.
- We derive a new numerically stable form of decoder preventing the numerical instability of the neural network.
- We design a computationally efficient subspace clustering cost to find both latent representation and a latent affinity matrix simultaneously for image clustering tasks.

In experiments, the validity of the proposed components is shown by doing ablation experiments on our architecture and cost function. Also, the superior performance of the proposed method is validated by comparing it with the

state-of-the-art methods and visualizing the graph clustered by our framework.

2. Preliminaries

2.1. Basic notations on graphs

A graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$, where \mathcal{V} denotes the node set with $v_i \in \mathcal{V}$ and $|\mathcal{V}| = n$, \mathcal{E} denotes the edge set with $(v_i, v_j) \in \mathcal{E}$, and $A \in \mathbb{R}^{n \times n}$ denotes an affinity matrix which encodes pairwise affinities between nodes. $D \in \mathbb{R}^{n \times n}$ denotes a degree matrix which is a diagonal matrix with $D_{ii} = \sum_j A_{ij}$. Unnormalized graph Laplacian Δ is defined by $\Delta = D - A$ [5]. Symmetric graph Laplacian L and random walk graph Laplacian L_{rw} are defined by $L = I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ and $L_{rw} = I_n - D^{-1}A$ respectively, where $I_n \in \mathbb{R}^{n \times n}$ denotes an identity matrix. Note that the Δ , L and L_{rw} are positive semidefinite matrices.

2.2. Spectral convolution on graphs

A spectral convolution on a graph [31] is the multiplication of an input signal $x \in \mathbb{R}^n$ with a spectral filter $g_\theta = diag(\theta)$ parameterized by the vector of Fourier coefficients $\theta \in \mathbb{R}^n$ as follows:

$$g_\theta * x = U g_\theta U^T x, \quad (1)$$

where U is the matrix of eigenvectors of the symmetric graph Laplacian $L = U \Lambda U^T$. $U^T x$ is the graph Fourier transform of the input x , and g_θ is a function of the eigenvalues of L , i.e., $g_\theta(\Lambda)$, where Λ is the diagonal matrix of eigenvalues of L . However, this operation is inappropriate for large-scale graphs since it requires an eigendecomposition to obtain the eigenvalues and eigenvectors of L . To avoid computationally expensive operations, the spectral filter $g_\theta(\Lambda)$ was approximated by K^{th} order Chebyshev polynomials in previous works [10]. By doing so, the spectral convolution on the graph can be approximated as

$$g_\theta * x \approx U \sum_{k=0}^K \theta'_k T_k(\tilde{\Lambda}) U^T x = \sum_{k=0}^K \theta'_k T_k(\tilde{L}) x, \quad (2)$$

where $T_k(\cdot)$ and θ' denote the Chebyshev polynomials and a vector of the Chebyshev coefficients respectively. $\tilde{\Lambda}$ is $\frac{2}{\lambda_{max}} \Lambda - I_n$, λ_{max} denotes the largest eigenvalue of L and \tilde{L} is $U \tilde{\Lambda} U^T = \frac{2}{\lambda_{max}} L - I_n$. The approximated model above is used as a building block of a convolution on graphs in [6].

In the GCN [14], the Chebyshev approximation model was simplified by setting $K = 1$, $\lambda_{max} \approx 2$ and $\theta = \theta'_0 = -\theta'_1$. This makes the spectral convolution simplified as follows:

$$g_\theta * x \approx \theta(I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x. \quad (3)$$

However, repeated application of $I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ can cause numerical instabilities in neural networks since the

spectral radius of $I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is 2, and the Chebyshev polynomials form an orthonormal basis when its spectral radius is 1. To circumvent this issue, the GCN uses renormalization trick:

$$I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}, \quad (4)$$

where $\tilde{A} = A + I_n$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Since adding self-loop on nodes to an affinity matrix cannot affect the spectral radius of the corresponding graph Laplacian matrix [9], this renormalization trick can provide a numerically stable form of $I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ while maintaining the meaning of each elements as follows:

$$(I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})_{ij} = \begin{cases} 1 & i = j \\ A_{ij}/\sqrt{D_{ii}D_{jj}} & i \neq j \end{cases} \quad (5)$$

$$(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}})_{ij} = \begin{cases} 1/(D_{ii} + 1) & i = j \\ A_{ij}/\sqrt{(D_{ii} + 1)(D_{jj} + 1)} & i \neq j. \end{cases} \quad (6)$$

Finally, the forward-path of the GCN can be expressed by

$$H^{(m+1)} = \xi(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(m)}\Theta^{(m)}), \quad (7)$$

where $H^{(m)}$ is the activation matrix in the m^{th} layer and $H^{(0)}$ is the input nodes' feature matrix X . $\xi(\cdot)$ is a nonlinear activation function like $\text{ReLU}(\cdot) = \max(0, \cdot)$, and $\Theta^{(m)}$ is a trainable weight matrix. The GCN presents a computationally efficient convolutional process (given the assumption that \tilde{A} is sparse) and achieves an improved accuracy over the state-of-the-art methods in semi-supervised node classification task by using features of nodes and geometric structure of graph simultaneously.

2.3. Laplacian smoothing

Li et al. [19] demystify GCN [14] and show that GCN is a special form of Laplacian smoothing [32]. Laplacian smoothing is a process that calculates a new representation of the input as a weighted local average of its neighbors and itself. When we add a self-loop on the nodes, the affinity matrix becomes $\tilde{A} = A + I_n$ and the degree matrix becomes $\tilde{D} = D + I_n$. Then, the Laplacian smoothing equation is given as follows:

$$x_i^{(m+1)} = (1 - \gamma)x_i^{(m)} + \gamma \sum_j \frac{\tilde{A}_{ij}}{\tilde{D}_{ii}} x_j^{(m)}, \quad (8)$$

where $x_i^{(m+1)}$ is the new representation of $x_i^{(m)}$, and γ ($0 < \gamma \leq 1$) is a regularization parameter which controls the importance between itself and its neighbors. We can rewrite the above equation in a matrix form as follows:

$$\begin{aligned} X^{(m+1)} &= (1 - \gamma)X^{(m)} + \gamma \tilde{D}^{-1}\tilde{A}X^{(m)} \\ &= X^{(m)} - \gamma(I_n - \tilde{D}^{-1}\tilde{A})X^{(m)} \\ &= X^{(m)} - \gamma \tilde{L}_{rw}X^{(m)}. \end{aligned} \quad (9)$$

If we set $\gamma = 1$ and replace \tilde{L}_{rw} with \tilde{L} , then Eq. (9) is changed into $X^{(m+1)} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X^{(m)}$ and this equation is the same as the renormalized version of spectral convolution in Eq. (7). From the above interpretation, Li et al. explain that the superior performance of GCN in semi-supervised node classification task is due to Laplacian smoothing which makes the features of nodes in the same clusters become similar.

3. The proposed method

In this section, we propose a novel graph convolutional autoencoder framework, named as GALA (Graph convolutional Autoencoder using LAplacian smoothing and sharpening). In GALA, there are M layers in total, from the first to $\frac{M}{2}$ th layers for the encoder and from the $(\frac{M}{2} + 1)$ th to M th layers for the decoder where M is an even number. The encoder part of GALA is designed to perform the computationally efficient spectral convolution on the graph with a numerically stable form of Laplacian smoothing in the Eq. (7) [14]. Along with this, its decoder part is designed to be a special form of Laplacian sharpening [32], unlike the existing VGAE-related algorithms. By this decoder part, GALA reconstructs the feature matrix of nodes directly, instead of yielding an affinity matrix as in the existing VGAE-related algorithms whose decoder parts are incomplete. Furthermore, to enhance the performance of image clustering, we devise a computationally efficient subspace clustering cost term which is added to the reconstruction cost of GALA.

3.1. Laplacian sharpening

Because the encoder performs Laplacian smoothing that makes the latent representation of each node similar to those of its neighboring nodes, we design the decoder part to perform Laplacian sharpening as the counterpart of Laplacian smoothing. Laplacian sharpening is a process that makes the reconstructed feature of each node farther away from the centroid of its neighbors, which accelerates the reconstruction along with the reconstruction cost and is governed by

$$x_i^{(m+1)} = (1 + \gamma)x_i^{(m)} - \gamma \sum_j \frac{A_{ij}}{D_{ii}} x_j^{(m)}, \quad (10)$$

where $x_i^{(m+1)}$ is the new representation of $x_i^{(m)}$, and γ is the regularization parameter which controls the importance between itself and its neighbors. The matrix form of Eq. (10) is given by

$$\begin{aligned} X^{(m+1)} &= (1 + \gamma)X^{(m)} - \gamma D^{-1}AX^{(m)} \\ &= X^{(m)} + \gamma(I_n - D^{-1}A)X^{(m)} \\ &= X^{(m)} + \gamma L_{rw}X^{(m)}. \end{aligned} \quad (11)$$

Analogous to the encoder, we set $\gamma = 1$ and replace L_{rw} with L . Similar to Eq. (3), we can express Laplacian sharp-

ening in the form of Chebyshev polynomial and simplify it with $K = 1$, $\lambda_{max} \approx 2$, and $\theta = \frac{1}{2}\theta'_0 = \theta'_1$. Then, a decoder layer can be expressed by

$$H^{(m+1)} = \xi((2I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}})H^{(m)}\Theta^{(m)}), \quad (12)$$

where $H^{(m)}$ is the matrix of the activation in the m^{th} layer, $2I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is a special form of Laplacian sharpening, $\xi(\cdot)$ is the nonlinear activation function like ReLU(\cdot) = $\max(0, \cdot)$, and $\Theta^{(m)}$ is a trainable weight matrix. However, since the spectral radius of $2I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is 3, repeated application of this operator can be numerically unstable. Hence, as GCN finds a numerically stable form of Chebyshev polynomials, we have to find a numerically stable form of Laplacian sharpening while maintaining its meaning.

3.2. Numerically stable Laplacian sharpening

To find a new representation of Laplacian sharpening whose spectral radius is 1, we use a signed graph [18]. A signed graph is denoted by $\Gamma = (\mathcal{V}, \mathcal{E}, \hat{A})$ which is induced from the unsigned graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$, where each element in \hat{A} has the same absolute value with A , but its sign is changed into minus or keeps plus. The degree matrix of the signed graph Γ is denoted by \hat{D} which is obtained from \hat{A} . In the signed graph, a problem occurs when calculating the degree matrix \hat{D} by the conventional way that may cancel the mixed signed weights in summation and so fails to yield the degree value representing the connectivity of a node to its neighbors. Thus, by following the practice for signed graphs, we calculate the degree of each node by $\hat{D}_{ii} = \sum_j |\hat{A}_{ij}|$ that has the same value (degree of connectivity) as in the unsigned graph. By using \hat{A} and \hat{D} , we can construct an unnormalized graph Laplacian $\hat{\Delta} = \hat{D} - \hat{A}$ and symmetric graph Laplacian $\hat{L} = I_n - \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ of the signed graph. From Theorem 1 of [18], the range of the eigenvalue of \hat{L} is $[0, 2]$, thus the spectral radius of $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ is 1 for any choice of \hat{A} . Using this result, instead of Eq. (12), we use a numerically stable form of Laplacian sharpening with spectral radius of 1, given by

$$H^{(m+1)} = \xi(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(m)}\Theta^{(m)}). \quad (13)$$

The remaining issue is to choose \hat{A} induced from A so that $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ maintains the meaning of each element of $2I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ in Eq. (12). To achieve this, we map all weights of the unsigned A to negative weights and adding a self-loop with a weight value 2 to each node, that is, $\hat{A} = 2I_n - A$ and $\hat{D} = 2I_n + D$. Then, each element of $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ is obtained by

$$(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}})_{ij} = \begin{cases} 2/(D_{ii} + 2) & i = j \\ -A_{ij}/\sqrt{(D_{ii} + 2)(D_{jj} + 2)} & i \neq j, \end{cases} \quad (14)$$

Table 1: Effectiveness of various decoders

	Cora			Citeseer		
	ACC	NMI	ARI	ACC	NMI	ARI
Eq. (7)	0.5628	0.4074	0.3289	0.5296	0.2588	0.2437
Eq. (12)	0.5999	0.4274	0.3775	0.5915	0.3177	0.3126
Eq. (16)	0.7459	0.5767	0.5315	0.6932	0.4411	0.4460

which has the same meaning with the original one given by

$$(2I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}})_{ij} = \begin{cases} 2 & i = j \\ -A_{ij}/\sqrt{D_{ii}D_{jj}} & i \neq j. \end{cases} \quad (15)$$

From Eqs. (13), (14) and (15), the numerically stable decoder layer of GALA is given as

$$H^{(m+1)} = \xi(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(m)}\Theta^{(m)}), (m = \frac{M}{2}, \dots, M-1), \quad (16)$$

where $\hat{A} = 2I_n - A$ and $\hat{D} = 2I_n + D$. The encoder part of GALA is constructed by using Eq. (7) as in GCN [14] as

$$H^{(m+1)} = \xi(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(m)}\Theta^{(m)}), (m = 0, \dots, \frac{M}{2}-1), \quad (17)$$

where $H^{(0)} = X$ is the feature matrix of the input nodes, $\tilde{A} = I_n + A$ and $\tilde{D} = I_n + D$. The complexity of propagation functions, Eqs. (16) and (17), are both $\mathcal{O}(mpc)$, where m is the cardinality of edges in the graph, p is the feature dimension of the previous layer, and c is the feature dimension of the current layer. Since the complexity is linear in the number of edges in the graph, the proposed algorithm is computationally efficient (given the assumption that A is sparse). Also, from Eq. (17), since GALA decodes the latent representation using both the graph structure and node features, the enhanced decoder of GALA can help to find more distinct latent representation.

In Table 1, we show the reason why the Laplacian smoothing is not appropriate to the decoder and the necessity of numerically stable Laplacian sharpening by node clustering experiments (the higher values imply the more correct results). Laplacian smoothing decoder (Eq. 7) shows the lowest performances, since Laplacian smoothing which makes the representation of each node similar to those of its neighboring nodes conflicts with the purpose of reconstruction cost. A numerically unstable form of Laplacian sharpening decoder (Eq. 12) shows higher performance compared to smoothing decoder because the role of Laplacian sharpening coincide with reconstructing the node feature. The performance of proposed numerically stable Laplacian sharpening decoder (Eq. 16) significantly higher than others, since it solves instability issue of neural network while maintaining the meaning of original Laplacian sharpening.

The basic cost function of GALA is given by

$$\min_{\bar{X}} \frac{1}{2} \|X - \bar{X}\|_F^2, \quad (18)$$

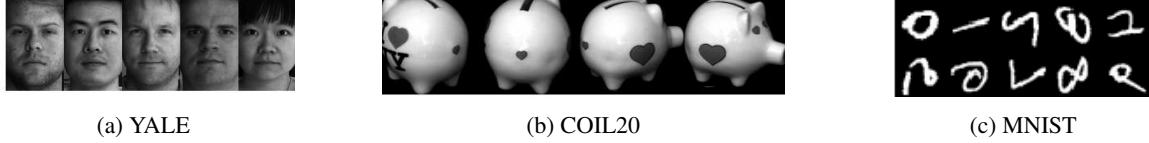


Figure 2: Sample images of three image datasets

where \bar{X} is the reconstructed feature matrix of nodes, the column of \bar{X} corresponds to the output of the decoder for an input feature of a node, and $\|\cdot\|_F$ denotes the Frobenius norm.

3.3. Subspace clustering cost for image clustering

It is a well-known assumption that image datasets are often drawn from multiple low-dimensional subspaces, although their data dimensions are high. Accordingly, subspace clustering, which has such an assumption about the input data in its own definition, has shown prominent clustering performance on various image datasets. Hence, we add an element of subspace clustering to the proposed method in the case of image clustering tasks. Among the various subspace clustering models, we add Least Squares Regression (LSR) [22] model for computational efficiency. Then the cost function for training of GALA becomes

$$\min_{\bar{X}, H, A_H} \frac{1}{2} \|X - \bar{X}\|_F^2 + \frac{\lambda}{2} \|H - HA_H\|_F^2 + \frac{\mu}{2} \|A_H\|_F^2, \quad (19)$$

where $H \in \mathbb{R}^{k \times n}$ denotes the latent representations (i.e., the output of the encoder), $A_H \in \mathbb{R}^{n \times n}$ denotes the affinity matrix which is a new latent variable for subspace clustering, and λ, μ are the regularization parameters. The second term of Eq. (19) aims at the self-expressive model of subspace clustering and the third term of Eq. (19) is for regularizing A_H . If we only consider minimizing A_H , the problem becomes:

$$\min_{A_H} \frac{\lambda}{2} \|H - HA_H\|_F^2 + \frac{\mu}{2} \|A_H\|_F^2. \quad (20)$$

We can easily obtain the analytic solution $A_H^* = (\frac{\mu}{\lambda} I_n)^{-1} H^T H$ by the fact that LSR model is quadratic on the variable A_H . By using this analytic solution and singular value decomposition, we derive a computationally efficient subspace clustering cost function as follows (The details are reported in the supplementary material):

$$\min_{\bar{X}, H} \frac{1}{2} \|X - \bar{X}\|_F^2 + \frac{\mu\lambda}{2} \text{tr}((\mu I_k + \lambda H H^T)^{-1} H H^T), \quad (21)$$

where $\text{tr}(\cdot)$ denotes the trace of the matrix. The above problem can be solved by $k \times k$ matrix inversion instead of $n \times n$ matrix inversion. Since the dimension of latent representation (k) is much smaller than the number of nodes (n), this simplification can reduce the computational burden significantly from $\mathcal{O}(n^3)$ to $\mathcal{O}(k^3)$.

3.4. Training

We train GALA to minimize Eq. (18) by using the ADAM algorithm [12]. We train GALA deterministically by using the full batch in each training epoch and stop when the cost is converged, so the number of epochs of each dataset varies. Note here that using the full batch during training is a common approach in neural networks based on spectral convolution on graph. Specifically, we set the learning rate to 1.0×10^{-4} for training and train GALA in an unsupervised way without any cluster labels. When the subspace clustering cost is added to reconstruction cost for image clustering tasks, we use pre-training and fine-tuning strategies similar to the ones in [11] to train GALA. First, in the pre-training stage, the training method is the same as that of minimizing Eq. (18). After pre-training, we fine-tune GALA to minimize Eq. (21) using ADAM. As in the pre-training, we train GALA deterministically by using full batch in each training epoch, and we set the number of epochs of the fine-tuning stage as 50 for all dataset. We set the learning rate to 1.0×10^{-6} for fine-tuning.

After the training process are over, we construct k -nearest neighborhood graph using attained latent representations H^* . Then we perform spectral clustering [26] and get the clustering performance. In the case of image clustering, after all training processes are over, we construct the optimal affinity matrix A_H^* noted in the previous subsection by using the attained latent representation matrix H^* from GALA. Then we perform spectral clustering [26] on the affinity matrix and get the optimal clustering with respect to our cost function.

4. Experiments

4.1. Datasets

We use four network datasets (Cora, Citeseer, Wiki, and Pubmed) and three image datasets (COIL20, YALE, and MNIST) for node clustering and link prediction tasks. Every network dataset has the feature matrix X and the affinity matrix A and every image dataset has the feature matrix X only. The summary of each dataset are presented in Table 3 and details are reported in the supplementary material. Also, the sample images of each image dataset are described in Figure 2.

4.2. Experimental settings

To measure the performance of node clustering task, we use three metrics: accuracy (ACC), normalized mutual in-

Table 2: Experimental results of node clustering

	Cora			Citeseer			Wiki		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
Kmeans[21]	0.4922	0.3210	0.2296	0.5401	0.3054	0.2786	0.4172	0.4402	0.1507
Spectral[26]	0.3672	0.1267	0.0311	0.2389	0.0557	0.0100	0.2204	0.1817	0.0146
Big-Clam[38]	0.2718	0.0073	0.0011	0.2500	0.0357	0.0071	0.1563	0.0900	0.0070
DeepWalk[28]	0.4840	0.3270	0.2427	0.3365	0.0878	0.0922	0.3846	0.3238	0.1703
GraEnc[33]	0.3249	0.1093	0.0055	0.2252	0.0330	0.0100	0.2067	0.1207	0.0049
DNGR[3]	0.4191	0.3184	0.1422	0.3259	0.1802	0.0429	0.3758	0.3585	0.1797
Circles[17]	0.6067	0.4042	0.3620	0.5716	0.3007	0.2930	0.4241	0.4180	0.2420
RTM[4]	0.4396	0.2301	0.1691	0.4509	0.2393	0.2026	0.4364	0.4495	0.1384
RMSC[36]	0.4066	0.2551	0.0895	0.2950	0.1387	0.0488	0.3976	0.4150	0.1116
TADW[37]	0.5603	0.4411	0.3320	0.4548	0.2914	0.2281	0.3096	0.2713	0.0454
GALA	0.7459	0.5767	0.5315	0.6932	0.4411	0.4460	0.5447	0.5036	0.3888

Table 3: Summary of datasets

	# Nodes	Dimension	Classes	# Edges
Cora[29]	2708	1433	7	5429
Citeseer[29]	3312	3703	6	4732
Wiki[37]	2405	4973	17	17981
Pubmed[29]	19717	500	3	44338
COIL20[25]	1440	1024	20	—
YALE[8]	5850	1200	10	—
MNIST[16]	10000	784	10	—

formation (NMI), and adjusted rand index (ARI) as in [35]. We report the mean values of the three metrics for each algorithm after executing 50 times, and the higher values imply the more correct results. For link prediction task, we partitioned the dataset following the work of GAE [13], and reported mean scores and standard errors of Area Under Curve (AUC) and Average Precision (AP) with 10 random initializations. The implementation details such as hyperparameters are reported in the supplementary material.

4.3. Comparing methods

We compare the performance of 15 algorithms. Compared algorithms can be categorized into four groups as described below:

- **i) Using features only.** ‘Kmeans’ [21] is the K-means clustering based on only the features of the data, which is the baseline clustering algorithm in our experiment.
- **ii) Using network structures only.** ‘Spectral’ [26] is a spectral clustering algorithm using eigendecomposition on graph Laplacian. ‘Big-Clam’ [38] is a large-scale community detection algorithm utilizing a variant of non-negative matrix factorization. ‘DeepWalk’ [28] learns the latent social representation of nodes using local information through a neural network. ‘GraEnc’ [33] is a graph-encoding neural network derived from the relation be-

tween autoencoder and spectral clustering. ‘DNGR’ [3] generates a low-dimensional representation of each node by using a graph structure and a stacked denoised autoencoder.

- **iii) Using both.** ‘Circles’ [17] is an algorithm which discovers social circles through a node clustering algorithm. ‘RTM’ [4] presents a relational topic model of documents and links between the documents. ‘RMSC’ [36] is a robust multi-view spectral clustering algorithm which can handle noises in the data and recover transition matrix through low-rank and sparse decomposition. ‘TADW’ [37] interprets DeepWalk from the view of matrix factorization and incorporates text features of nodes.
- **iv) Using both with spectral convolution on graphs.** ‘GAE’ [13] is the first attempt to graft the spectral convolution on graphs onto autoencoder framework. ‘VGAE’ [13] is the variational variant of GAE. ‘MGAE’ [35] is an autoencoder which combines the marginalization process with spectral convolution on graphs. ‘ARGA’ [27] learns the latent representation by adding an adversarial model to a non-probabilistic variant of VGAE. ‘ARVGA’ [27] is an algorithm which adds an adversarial model to VGAE.

4.4. Node clustering results

The experimental results of node clustering are presented in Table 2. It can be observed that for every dataset, the methods which use features and network structures simultaneously show better performance than the methods which use only one of them. Furthermore, among the methods which use both features and network structures, algorithms with neural network models which exploit spectral convolution on graphs present outstanding

Table 4: Experimental results of image clustering

	COIL20			YALE			MNIST		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
Kmeans[21]	0.6118	0.7541	0.5545	0.7450	0.8715	0.7394	0.5628	0.5450	0.4213
Spectral[26]	0.6806	0.8324	0.6190	0.5793	0.7202	0.4600	0.6496	0.7204	0.5836
GAE[13]	0.6632	0.7420	0.5514	0.8520	0.8851	0.8122	0.7043	0.6535	0.5534
VGAE[13]	0.6847	0.7465	0.5627	0.9157	0.9358	0.8873	0.7163	0.7149	0.6154
MGAE[35]	0.6507	0.7889	0.6004	0.8203	0.8550	0.7636	0.5807	0.5820	0.4362
ARGA[27]	0.7271	0.7895	0.6183	0.9309	0.9394	0.8961	0.6672	0.6759	0.5552
ARVGA[27]	0.7222	0.7917	0.6240	0.8727	0.8803	0.7944	0.6328	0.6123	0.4909
GALA	0.8000	0.8771	0.7550	0.8530	0.9486	0.8647	0.7384	0.7506	0.6469
GALA+SCC	0.8229	0.8851	0.7579	0.9933	0.9860	0.9854	0.7426	0.7565	0.6675

Table 5: Experiment results on Pubmed dataset

	ACC	NMI	ARI
Kmeans[21]	0.5952	0.3152	0.2817
Spectral[26]	0.5282	0.0971	0.0620
GAE[13]	0.6861	0.2957	0.3046
VGAE[13]	0.6887	0.3108	0.3018
MGAE[35]	0.5932	0.2822	0.2483
ARGA[27]	0.6807	0.2757	0.2910
ARVGA[27]	0.5130	0.1169	0.0777
GALA	0.6939	0.3273	0.3214

performance since they can learn deeper relationships between nodes than the methods which do not use spectral convolution on graphs. In every experiments, GALA shows superior performance to other methods. Especially, for the Cora dataset, GALA outperforms VGAE, which is the first graph convolution autoencoder framework, by about 24.39%, 24.75% and 27.68%, and MGAE, which is the state-of-the-art graph convolutional autoencoder algorithm, by about 6.15%, 6.56% and 8.68% on ACC, NMI and ARI, respectively. The better performance of GALA comes from the better decoder design based on the numerically stable form of Laplacian sharpening both and full utilizing of graph structure and node attributes in the whole autoencoder framework.

Furthermore, we conduct another node clustering experiment on a large network dataset (Pubmed), and the results are reported in Table 5. We can observe that GALA outperforms every baselines and state-of-the-art graph convolution algorithms. Although Kmeans clustering, a baseline algorithm, shows higher performance over several graph convolution algorithms on NMI and ARI, the proposed method presents better performances.

4.5. Image clustering results

The experimental results of image clustering are presented in Table 4. We report both GALA’s performance of reconstruction cost only case and the subspace clustering cost added case (GALA+SCC). It can be seen that GALA outperforms several baselines and the state-of-the-art graph convolution algorithms for most of the cases. Also, for ev-

ery case, the proposed subspace clustering cost term contributes to improve the performance of the image clustering. On the YALE dataset, notably, we can observe that the proposed subspace clustering cost term significantly enhances the image clustering performance and achieves nearly perfect accuracy.

4.6. Ablation studies

We validate the effectiveness of the proposed stable decoder and the subspace clustering cost by image clustering experiments on the three image datasets (COIL20, YALE and MNIST). There are four configurations as shown in Table 6. We would like to note that the reconstruction cost only (Eq. 18) is a subset of subspace clustering cost (Eq. 21), thus the last configuration is the full proposed method. Reported numbers are mean values after executing 50 times. It can be clearly noticed that the numerically stable form of Laplacian sharpening and subspace clustering cost are helpful to find the latent representations which reflect the graph structures certainly and using both components can boost the performance of clustering. In addition, it can be seen that the stable decoder with the reconstruction cost only outperforms the state-of-the-art algorithms in most cases because GALA can utilize the graph structure in the whole processes of the autoencoder architecture.

4.7. Link prediction results

We provide some results on link prediction task on Cite-seer dataset. For link prediction task, we minimized the below cost function that added link prediction cost of GAE [13] to the reconstruction cost, where H is the latent representation, $\hat{A} = \text{sigmoid}(HH^T)$ is the reconstructed affinity matrix and γ is the regularization parameter.

$$\min_{\bar{X}, H} \frac{1}{2} \|X - \bar{X}\|_F^2 + \gamma \mathbb{E}_H [\log p(\hat{A}|H)]. \quad (22)$$

The results are shown in Table 7, and our model outperforms the compared methods in terms of the link prediction task as well as the node clustering task.

Table 6: Effects of stable decoder and subspace clustering cost

	COIL20			YALE			MNIST		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
Unstable decoder and reconstruction cost only (Eq. 12 and Eq. 18)	0.5961	0.7986	0.5492	0.7205	0.9028	0.7530	0.6589	0.7397	0.5983
Unstable decoder and subspace clustering cost (Eq. 12 and Eq. 21)	0.7104	0.8074	0.6429	0.7810	0.8710	0.7130	0.6734	0.7211	0.6028
Stable decoder and reconstruction cost only (Eq. 16 and Eq. 18)	0.8000	0.8771	0.7550	0.8530	0.9486	0.8646	0.7384	0.7506	0.6469
Stable decoder and subspace clustering cost (Eq. 16 and Eq. 21)	0.8229	0.8851	0.7579	0.9933	0.9860	0.9854	0.7426	0.7565	0.6675

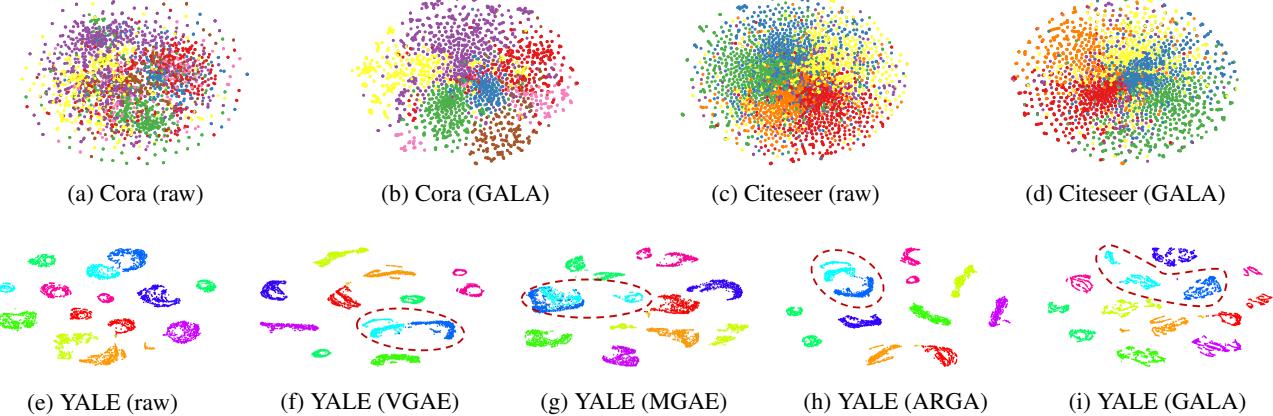


Figure 3: The two-dimensional visualizations of raw features of each node and the latent representations of compared methods and GALA for Cora, Citeseer and YALE are presented. The same color indicates the same cluster.

Table 7: Experimental results of link prediction on Citeseer

	AUC	AP
GAE[13]	89.5 ± 0.04	89.9 ± 0.05
VGAE[13]	90.8 ± 0.02	92.0 ± 0.02
ARGA[27]	91.9 ± 0.003	93.0 ± 0.003
ARVGA[27]	92.4 ± 0.003	93.0 ± 0.003
GALA	94.4 ± 0.009	94.8 ± 0.010

4.8. Visualization

One of the key ideas of the proposed autoencoder is that the encoder makes the feature of each node becomes similar with its neighbors, and the decoder makes the features of each node distinguishable with its neighbors using the geometrical structure of the graphs. To validate the proposed model, we visualize the distribution of learned latent representations and the input features of each node in two-dimensional space using *t*-SNE [23] as shown in Figure 3. From the visualization, we can see that GALA is well-clustering the data according to their corresponding labels even though GALA performs in an unsupervised manner. Also, we can see through the red dotted line in embedding results of the latent representation on YALE that GALA embeds the representation of nodes better than the compared methods by minimizing inter-cluster affinity and maximiz-

ing intra-cluster affinity.

5. Conclusions

In this paper, we proposed a novel autoencoder framework which can extract low-dimensional latent representations from a graph in irregular domains. We designed a symmetric graph convolutional autoencoder architecture where the encoder performs Laplacian smoothing while the decoder performs Laplacian sharpening. Also, to prevent numerical instabilities, we designed a new representation of Laplacian sharpening with spectral radius one by incorporating the concept of the signed graph. To enhance the performance of image clustering tasks, we added a subspace clustering cost term to the reconstruction cost of the autoencoder. Experimental results on the network and image datasets demonstrated the validity of the proposed framework and had shown superior performance over various graph-based clustering algorithms.

Acknowledgement: This work was supported by Next Generation ICD Program through NRF funded by Ministry of S&ICT [2017M3C4A7077582], ICT R&D program of MSIP/IITP [No.B0101-15-0552, Predictive Visual Intelligence Technology], and the Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Science and ICT under Grant NRF-2017R1C1B2012277.

References

- [1] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, pages 585–591, 2002. 1
- [2] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 1
- [3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *AAAI*, pages 1145–1152, 2016. 6
- [4] Jonathan Chang and David Blei. Relational topic models for document networks. In *Artificial Intelligence and Statistics*, pages 81–88, 2009. 6
- [5] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997. 2
- [6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016. 2
- [7] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pages 2224–2232, 2015. 1
- [8] Athinodoros S Georghiades, Peter N Belhumeur, and David J Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):643–660, 2001. 6
- [9] Rostislav I Grigorchuk and Andrzej Zuk. On the asymptotic spectrum of random walks on infinite families of graphs. *Random Walks and Discrete Potential Theory (Cortona, 1997), Sympos. Math.*, 39:188–204, 1999. 3
- [10] David K Hammond, Pierre Vandergheynst, and Rémi Grisonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011. 2
- [11] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. Deep subspace clustering networks. In *Advances in Neural Information Processing Systems*, pages 24–33, 2017. 2, 5
- [12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5
- [13] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016. 1, 2, 6, 7, 8
- [14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. 1, 2, 3, 4
- [15] David Lazer, Alex Sandy Pentland, Lada Adamic, Sinan Aral, Albert Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, et al. Life in the network: the coming age of computational social science. *Science (New York, NY)*, 323(5915):721, 2009. 1
- [16] Yann LeCun. The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>, 1998. 6
- [17] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in Neural Information Processing Systems*, pages 539–547, 2012. 6
- [18] Hong Hai Li and Jiong Sheng Li. Note on the normalized laplacian eigenvalues of signed graphs. *Australas. J. Combin.*, 44:153–162, 2009. 2, 4
- [19] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2, 3
- [20] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. *arXiv preprint arXiv:1712.00268*, 2017. 1
- [21] Stuart Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. 6, 7
- [22] Can-Yi Lu, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang, and Shuicheng Yan. Robust and efficient subspace segmentation via least squares regression. In *European conference on computer vision*, pages 347–360. Springer, 2012. 5
- [23] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 8
- [24] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems*, pages 3697–3707, 2017. 1
- [25] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). 1996. 6
- [26] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2002. 1, 5, 6, 7
- [27] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*, pages 2609–2615, 2018. 1, 6, 7, 8
- [28] 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. ACM, 2014. 6
- [29] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93, 2008. 6
- [30] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 1
- [31] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data

- analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013. 2
- [32] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd Annual Conference on Computer graphics and Interactive techniques*, pages 351–358. ACM, 1995. 2, 3
- [33] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *AAAI*, pages 1293–1299, 2014. 6
- [34] René Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011. 2
- [35] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 889–898. ACM, 2017. 1, 6, 7
- [36] Rongkai Xia, Yan Pan, Lei Du, and Jian Yin. Robust multi-view spectral clustering via low-rank and sparse decomposition. In *AAAI*, pages 2149–2155, 2014. 6
- [37] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *IJCAI*, pages 2111–2117, 2015. 6
- [38] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM International Conference on Web Search and Data Mining*, pages 587–596. ACM, 2013. 6
- [39] Pan Zhou, Yunqing Hou, and Jiashi Feng. Deep adversarial subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1596–1604, 2018. 2