

# Embedding Graphs on Grassmann Manifold

Bingxin Zhou<sup>a,b,1,\*</sup>, Xuebin Zheng<sup>a,1</sup>, Yu Guang Wang<sup>b,c</sup>, Ming Li<sup>d</sup>, Junbin Gao<sup>a</sup>

<sup>a</sup>*The University of Sydney Business School, The University of Sydney, NSW, Australia*

<sup>b</sup>*Institute of Natural Sciences and School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai, China*

<sup>c</sup>*School of Mathematics and Statistics, The University of New South Wales, Australia.*

<sup>d</sup>*Key Laboratory of Intelligent Education Technology and Application of Zhejiang Province, Zhejiang Normal University, China*

---

## Abstract

Learning efficient graph representation is the key to favorably addressing downstream tasks on graphs, such as node or graph property prediction. Given the non-Euclidean structural property of graphs, preserving the original graph data's similarity relationship in the embedded space needs specific tools and a similarity metric. This paper develops a new graph representation learning scheme, namely EGG, which embeds approximated second-order graph characteristics into a Grassmann manifold. The proposed strategy leverages graph convolutions to learn hidden representations of the corresponding subspace of the graph, which is then mapped to a Grassmann point of a low dimensional manifold through truncated singular value decomposition (SVD). The established graph embedding approximates denoised correlation of node attributes, as implemented in the form of a symmetric matrix space for Euclidean calculation. The effectiveness of EGG is demonstrated using both clustering and classification tasks at the node level and graph level. It outperforms baseline models on various benchmarks.

**Keywords:** Grassmann Manifold, Graph Neural Network, Projection Embedding, Subspace Clustering

---

---

\*Corresponding author

Email address: [bzho3923@uni.sydney.edu.au](mailto:bzho3923@uni.sydney.edu.au) (Bingxin Zhou)

<sup>1</sup>Both authors contribute equally

## 1. Introduction

Graph neural networks (GNNs), as one of the most prominent avenues in geometric deep learning, have received growing attention over the last few years [1, 2, 3, 4, 5]. Common to many GNN-based predictive tasks, distilling key features and structural information from the given graph data stays within the core of designing an effective graph representation learning.

Graph convolution [6], especially graph neural message passing [7], provides an efficient expression to the information flow of the underlying graph through aggregating regional features over the node neighborhood. For a set of multiple graphs of varying size and topological structure, employing arbitrary graph convolutions fails to coincide the size of graph representation. Instead, an appropriate graph pooling scheme is required to establish graph-level representations of a uniform scale. Furthermore, a handful of pooling strategies has been proposed to scale down the graph embedding by extracting the key components of the graph representation. Depending on whether the hidden attributes are coarsened along the adjacency matrix, different strategies are categorized as either global [8, 9, 10] or hierarchical pooling [11, 12, 13].

While each operation designs its unique standard for graph coarsening and feature extraction, common to all pooling strategies is the requirement of node permutation invariance. When employing feature extraction, the node order of an undirected graph must not incline the network to excessively concentrate on individual attributes. The permutation invariance property is intuitive in heuristic pooling operations like summation, averaging, and maximization, where the aggregation rules of regional patterns do not rely on the arrangement of nodes sequence. One possibility to define a permutation invariant pooling operation on graph topology is to view nodes of a graph as a set of elements. For instance, the authors of [14] establish the Wasserstein embedding of a node set under the linear optimal transport framework [15]. A proper Wasserstein metric is designed to match pairs of node sets while avoiding a direct node selection criterion, as the hierarchical pooling schemes.

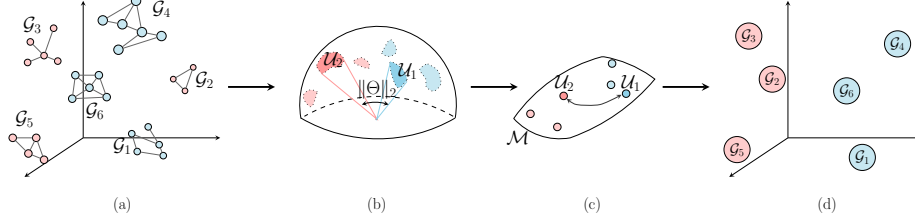


Figure 1: Computational principle of the proposed framework EGG. (a) Given graph  $\mathcal{G}_i$ , i.e., a set of nodes, the target is to train a model that assigns a label to each of them. (b) Every  $\mathcal{G}_i$  rectifies a subspace of its most representative bit on the manifold space, where their geodesic distance is measured by their principal angle. (c) From the perspective of orthonormal basis, these subspaces can be embedded to the Grassmann points of a Grassmann manifold, where similar points have a small distance. (d) These Grassmann points support an easy projection operation to the space of symmetric matrices for deep learning tasks, such as classification and clustering.

This work instead studies the expression of the node set of a graph through *manifold learning* [16]. Recall that the fundamental assumption of a smooth graph in the design of graph convolutions is that spatially connected nodes are likely to share similar characteristics and the nodes from the same class are likely to have similar attributes. Naturally, the hidden representation of an attributed graph creates a subspace of lower dimension, or equivalently, a point of a Grassmann manifold. Consequently, a sophisticated learning task over graphs of varying size and topology accomplishes its transformation to a new learning task over Grassmann points of a fixed-dimensional Grassmann manifold.

We name the above embedding strategy of mapping graphs to Grassmann points as **E**mboding **G**raphs on a **G**rassmann manifold, or EGG for short. This framework is a generic method to express graphs with a Grassmann manifold subspace analysis. As demonstrated in Figure 1, each representation of node sets establishes a subspace of indeterminate dimensions. EGG then embeds these subspaces to Grassmann points of a Grassmannian  $\mathcal{M}$ , where every point is explicitly represented by an orthonormal matrix. Furthermore, these Grassmann

points support an effortless inversion to the Euclidean space through symmetric projection, where new representations are of the same dimension, and they are ready for conventional graph classification or segmentation tasks.

In comparison to existing graph distilling strategies, this new architecture design is one practice of exploiting principal components of graphs from non-linear information transformation. Each node community formulates a subspace of coarsened and smoother higher-level expression. While conventional graph aggregation generally requires stacked convolution or pooling layers to allow non-linear propagation, EGG leverages the truncated singular value decomposition (trSVD) to directly compress the principal components and construct the smooth subspace. In addition, the projected results from these acquired graph Grassmann embeddings approximates the second-order covariance of node attributes, which gains more expressive power than typical first-order expression from the conventional aggregation and distilling operations. Moreover, the proposed EGG for graph-level tasks guarantees the critical property of permutation invariance, which is an essential requirement of a qualified graph pooling design, yet it has been ignored by many existing methods.

The preliminary idea of the developed framework EGG was first introduced in a workshop paper [17]. This extension provides abundant details for understanding the rationale and paradigm of the proposed graph embedding scheme. Furthermore, the embedding strategy is expanded from graph pooling applications to more general scenarios, where in this complete work we exploit the possibility of handling lower-level unsupervised learning tasks of node segmentation. Additional investigations are addressed to interpret the learned expression and avoid the black-box model design.

The rest of this paper is organized as follows. Section 2 reviews the previous literature on graph representation learning and Grassmann deep learning applications. Section 3 introduces the Grassmann geometry that is closely related. Section 4 details the two critical ingredients of analyzing subspaces in a Grassmannian. We then demonstrate our methods with two specific applications: graph classification and node clustering. The problems are formulated in

Section 5, and the empirical performances are reported in Section 6. Further investigations on the significance of EGG are addressed in Section 7.

## 2. Related Work

An attributed undirected graph is denoted as  $\mathcal{G}_i = (V_i, E_i, \mathbf{X}_i)$  of  $n_i := |V_i|$  nodes and  $|E_i|$  edges. The node is featured by  $\mathbf{X}_i \in \mathbb{R}^{n_i \times d}$ , and the (weighted) edges for the structure information are described by an adjacency matrix  $\mathbf{A} \in \mathbb{R}^{n_i \times n_i}$ . Since the graph topology provides additional information, *graph representation learning* aims at encoding such a structural expression to conventional vector representations for deep learning models that assign labels to instances. A node-level graph learning task assigns a label  $\{\mathbf{y}_i\}$  to each node of the graph  $\mathcal{G}_i$ , while a graph-level task finds a sequence of  $N$  labels  $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\}$  from a set of input graphs  $\mathbb{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_N\}$ . Depending on the nature of the assigned labels, the learning task can be categorized to either regression or classification.

*Spatial Graph Convolution.* The emerging development of graph neural networks (GNNs) generates enormous work for graph representation learning. Typically, the topological embedding is realized by graph convolutional layers. A spatial-based propagation rule [7] leverages proper feature extraction and aggregation from the central node’s local community or the neighborhood. The propagation rule can be designed as flexible as weighted average [18, 19, 20], concatenation [21, 22], learnable attention [23, 24, 25], or other adaptive choices [26, 27]. To allow a broader receptive field, multiple convolution layers are frequently stacked for multi-hop neighborhood aggregation. Nevertheless, the majority of the aggregation rules are carried out in the first-order space, which omits the second-order covariance information that can capture insightful non-linear relationships of the feature attributes [28, 29].

*Graph Pooling and Down-sampling.* As a graph-level learning task involves multiple graphs of a diverse number of nodes, it is crucial for GNNs to unify the dimension of the output graph representation by pooling operations. For instance,

TOPKPOOL [30] formulates a score function to rank graph nodes and picks the parent nodes of subgraphs from graph clusters to hierarchically coarsen a graph. Other research enhances the selection efficacy through a carefully designed scoring mechanism, such as multilayer perceptron (MLP) [31] or attention [32, 33]. The authors of [11, 12, 13] extended graph coarsening conventions to different slicing principles, although the role of graph clustering and its influence on local pooling has been challenged by the literature [8, 34]. Alternatively, global graph pooling strategies are pumped out in practice with a simpler design and comparable performance [9, 10].

*Grassmann Manifold in Deep Learning.* Grassmann manifolds play an important role in recommender systems [35, 36], computer vision [37, 38] and pattern recognition [39, 40]. Grassmann learning exploits subspace-invariant features and harnesses the structural information of sample sets, which improves the prediction performance of a model with lower complexity and higher robustness. Due to these privileges, the Grassmann manifold is often approached as a tool of nonlinear dimensionality reduction [41, 42, 43, 44] or optimization objectives [45, 46]. While direct computations on a Grassmannian can be sophisticated, other research investigates pipeline Grassmann points to a Grassmann learning algorithm, such as Deep Grassmann Networks [47] and Grassmann clustering [48, 49].

### 3. Grassmann Geometry

This section overviews the mathematical formulation of the Grassmann points and Grassmann manifold. We also discuss the measurement on the geodesic distance for Grassmann points, as well as their Euclidean counterparts.

#### 3.1. Grassmann Manifold

Grassmann manifold is a manifold of matrices of a specific rank. Each Grassmann point is represented by an orthonormal matrix, and it corresponds to a subspace of the underlying real Euclidean space. To say it precisely,

**Definition 1** (Grassmann Manifold [16]). *The Grassmann manifold  $\mathcal{M}(p, m)$  ( $p \leq m$ ) consists of all  $p$ -dimensional subspaces of the Euclidean space  $\mathbb{R}^m$ , i.e.,*

$$\mathcal{M}(p, m) = \{\mathcal{U} \subset \mathbb{R}^m : \mathcal{U} \text{ is a subspace, } \dim(\mathcal{U}) = p\}.$$

A particular Grassmann point  $\mathcal{U} \in \mathcal{M}(p, m)$  is identified by an orthonormal matrix  $\mathbf{U} \in \mathbb{R}^{m \times p}$ , which is an equivalence class of all rank- $p$  matrices that spans  $\mathcal{U}$ . That is,  $\mathcal{M}(p, m) = \{\text{span}(\mathbf{U}) : \mathbf{U} \in \mathbb{R}^{m \times p}, \mathbf{U}^\top \mathbf{U} = \mathbb{I}_p\}$ . Furthermore, this subspace is identified uniquely by a projector  $\Pi(\mathbf{U})$  on  $\mathcal{U}$ , such that  $\Pi(\mathbf{U}) = \mathbf{U}\mathbf{U}^\top$ . The Grassmann manifold is an abstract quotient manifold that one can represent in many ways, such as the Lie group theory [50, 51]. To best allow convenient algebraic calculation, this work constructs Grassmann points from the perspective of projection matrices [52].

### 3.2. Subspace Distance

The distance between two Grassmann points is measured differently from the conventional Euclidean metric due to the curvature of the Grassmannian. The *geodesic distance* is thus defined as the length of the shortest path along the manifold between two points, which is a function of the principal angles of the two subspaces or analogously the two Grassmann points, as we introduce now.

**Definition 2** (Principal angle). *Given two Grassmann points  $\mathcal{U}_1, \mathcal{U}_2 \in \mathcal{M}(p, m)$  and their orthonormal bases  $\mathbf{U}_1 = [(\mathbf{u}_1)_1, \dots, (\mathbf{u}_1)_p]$ ,  $\mathbf{U}_2 = [(\mathbf{u}_2)_1, \dots, (\mathbf{u}_2)_p] \in \mathbb{R}^{m \times p}$ , we define their principal angles  $0 \leq \theta_1 \leq \dots \leq \theta_p \leq \frac{\pi}{2}$  recursively by*

$$\begin{aligned} \cos(\theta_i) &= \max(\mathbf{u}_1)_i^\top (\mathbf{u}_2)_i \\ \text{s.t. } \|\mathbf{u}_1\|_2 &= \|\mathbf{u}_2\|_2 = 1, \quad (\mathbf{u}_1)_i^\top (\mathbf{u}_1)_j = (\mathbf{u}_2)_i^\top (\mathbf{u}_2)_j = 0 \quad \forall j < i. \end{aligned}$$

The principal angles describe the smallest  $p$  angles between all possible bases of the two  $p$ -dimensional subspaces ( $\mathbf{U}_1$  and  $\mathbf{U}_2$ ). With a sequence of principal angles  $\Theta = [\theta_1, \dots, \theta_p]$ , the geodesic distance between the two Grassmann points is a function of the principal angles, i.e.,  $d(\mathbf{U}_1, \mathbf{U}_2) = \|\Theta\|_2$ .

In literature, there exist many other measurements to describe the discrepancy between subspaces, so that a closed-form solution for optimization on Grassmann manifold becomes possible. For example, the *projection distance* [52] embeds a Grassmann manifold  $\mathcal{M}(p, m)$  into a higher  $m$  dimensional Euclidean space in the form of Symmetric Positive-Definite (SPD) matrix; the chordal distance and the Procrustes distance [53] measures the total squared sine angle between Grassmann points, which is usually used for shape analysis. This work follows the first measure of the projection distance, which also supports kernelized Grassmann learning [54] and has been well-explored in learning low-rank approximation [42, 43, 44] and pattern recognition [39, 40].

#### 4. Grassmannian Subspace Analysis on Graphs

This section provides a guideline for graph smoothing and distilling through node sets embedding to a continuous and smooth Grassmann manifold. As we shall introduce below, our proposed method rectifies graph representations with embedded structure information to a Grassmannian of their feature space at a lower dimension. The Euclidean representations of these Grassmann instances from the projection perspective can be considered as an approximate version of the feature correlations that eliminates unnecessary variances. A Grassmann embedding appends non-linear smoothing effects to the graph representations that are usually achieved by stacking up fully-connected layers. The output representation meets the key requirements of a standard graph embedding scheme, and it allows arbitrary computations adapted to a Euclidean space.

##### 4.1. Problem Formulation

We begin with a set of hidden representations  $\mathbb{H} = \{\mathbf{H}_1, \dots, \mathbf{H}_N\}$  from graph convolutional layers for a given set of  $N$  graphs  $\mathbb{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_N\}$ . Here  $\mathbf{H}_i \in \mathbb{R}^{n_i \times m}$  is with respect to  $n_i$  nodes in  $\mathcal{G}_i$  and a number of  $m$  hidden neurons for the last graph convolution operation. The  $\mathbf{H}_i$  for graph-level learning tasks requires a graph representation that is irrelevant to the node size and



has an identical dimension to other  $\mathbf{H} \in \mathbb{H}$ . Therefore, at the rectification step, each graph representation of  $\mathbb{H}$  to a Grassmann point is identified with an orthonormal basis of  $\mathbf{H}$ , and is aligned to the same Grassmannian.

#### 4.2. Manifold Rectification

Suppose we have obtained a graph hidden representation  $\mathbf{H} \in \mathbb{H} \subset \mathbb{R}^{n \times m}$  as well as its row-generated subspace  $\text{span}(\mathbf{H}^\top)$ , which as mentioned can be achieved by employing one or multiple layers of graph convolution. Our target is to find a concrete Grassmann representation with most variations of the data. We characterize the representation as an orthogonal basis of the subspace, which can be rectified in several ways, such as the QR decomposition of matrix  $\mathbf{H}^\top$ , as demonstrated in [47]. Here we consider another classic method of the truncated singular value decomposition (SVD) to find the best low-rank approximation of the hidden feature space  $\mathbf{H}^\top$  in the sense of the least-squares [52].

The preliminary goal of employing the manifold embedding is to establish a graph representation of a unite dimension  $k$ . We therefore leverage the truncated SVD on  $\mathbf{H}$  to obtain the most representative basis  $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p]$  of the subspace  $\text{span}(\mathbf{H}^\top)$ , i.e.,

$$\mathbf{H}^\top = \mathbf{U} \mathbf{S} \mathbf{V}^\top, \quad (1)$$

where the Grassmann point  $\mathcal{U} = [\mathbf{U}]$  is an equivalence class of  $\mathbf{U}$ . The  $\mathbf{U} \in \mathbb{R}^{m \times k}$  is an orthonormal basis with  $\text{rank}(\mathbf{H}^\top) = k, 1 \leq k \leq \min\{m, n\}$ . The diagonal  $\mathbf{S} := \text{Diag}([\sigma_1, \dots, \sigma_k]) \in \mathbb{R}^{k \times k}$  contains  $k$  singular values sorted in the descending order, where  $\sigma_l$  gives the percentage importance of  $\mathbf{u}_l$ . The corresponding singular vectors constitute  $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_k] \in \mathbb{R}^{n \times k}$ .

Rather than using the full  $\mathbf{U} \in \mathbb{R}^{m \times k}$ , we only preserve the first  $p$ -columns of  $\mathbf{U}$  ( $p \leq k$ ), denoted as  $\mathbf{U}_p$ , to include the most important  $p$  components of the original space  $\mathbf{H}$ . The subspace  $\text{span}(\mathbf{U}_p)$  composes a Grassmannian  $\mathcal{U} := [\mathbf{U}_p]$  in  $\mathcal{M}(p, m)$ . In practice, it could potentially hurt the expressiveness of graph embedding for defining an identical relatively small subspace dimension  $p$  for all graphs, since real-world datasets could have a great number of graphs with a large variation on node sizes from a few to thousands. Instead, we let  $p$  for

a Grassmann point  $[U_d] \in \mathcal{M}(p, m)$  be determined by  $p = \sum_{i=1}^k \mathbb{1}\{\sigma_i > r\}$ . The  $\sigma_i$  corresponds to unit singular values from (1), and  $r$  denotes the global threshold of the percentage importance. However, all these Grassmann points  $\{\mathcal{U}\}$  can be naturally mapped to the embedded Grassmannian  $\mathcal{M}(p_{\max}, m)$ , with  $p_{\max}$  the highest  $p$  over all Grassmann points, so they are still at the same space with accessible geodesic distance.

The embedding operation is compatible to arbitrary hidden representations of  $\mathbb{H}$ . Intuitively, it regards a (sub)graph with a set of  $n$  attributed nodes as a  $p$ -dimensional subspace. While a variant of size  $n$  can be observed from different node sets, the rectified Grassmann points from the underlying embedding operation are projected to the same Grassmann manifold, where the geodesic distance of two points reflects the similarity of two sets, or graph instances in analogue. Such similarity provides a criterion for distance-based training tasks, such as clustering or classification.

The rectification step by the truncated SVD can be considered as a non-linear transformation of the feature space that extracts the most powerful subspace expression of the node space and view it as a Grassmann point. At this stage, node sets of varying size are embedded to a common Grassmann manifold, where each of them is represented by a subspace of orthonormal basis  $U_p \in \mathbb{R}^{m \times p}$ . While it is feasible to compute the geodesic distance of Grassmann points, projecting them back to the Euclidean space is preferred by conventional deep learning modules. We now introduce the projection operation of a Grassmann point to its associated Euclidean representation.

#### 4.3. Projection Embedding

The rectification step establishes a set of Grassmann points  $\{\mathcal{U}_1, \dots, \mathcal{U}_N\} \subset \mathcal{M}(p, m)$  as well as their matrix representation  $\{U_1, \dots, U_N\}$  from the orthonormal basis perspective. One can establish follow-up learning schemes on  $\{\mathcal{U}_1, \dots, \mathcal{U}_N\}$  with the Grassmann geometry [16]. Alternatively, the Grassmann points can first be projected to the space of symmetric matrices  $\text{Sym}(m)$  to

---

**Algorithm 1:** EGG for graph representation learning

---

**Input** : Hidden representation  $\mathbf{H}$ , dimension  $p$ .

**Output:** A graph representation.

- 1 (for graph-level tasks): Transpose  $\mathbf{H}$  to  $\mathbf{H}^\top$ . // Transpose
  - 2 Find the  $p$ -dimensional low-rank representation of  $\mathcal{U}$  by (1).  
// Manifold Rectification
  - 3 Project  $\mathcal{U}$  to an SPD matrix (2).  
// Projection Embedding
- 

allow calculations based on Euclidean space. Define

$$\Pi : \mathcal{M}(p, m) \longrightarrow \text{Sym}(m), \quad \Pi(\mathbf{U}) = \mathbf{U}\mathbf{U}^\top. \quad (2)$$

The projected representation  $\text{Sym}(m)$  allows general Euclidean measures to allow conventional deep learning methods such as fully-connected layers.

After the rectification and the projection step, the graph latent representation  $\mathbf{H}$  is transformed to a symmetric positive definite (SPD) matrix  $\Pi(\mathbf{U}_p) = \mathbf{U}_p\mathbf{U}_p^\top$ . This SPD matrix representation is an analog to a bilinear mapping, and it captures the second-order statistics that better reflects regional features of  $\mathbf{H}$  [55]. Moreover, the rectified representation  $\mathbf{U}$  from (1) gains robustness as a result of approximating the covariance matrix  $\mathbf{U}_p\mathbf{U}_p^\top$ . This projected Euclidean representation is feasible for various tasks. For example, in graph property prediction, a vectorized  $\Pi(\mathbf{U}_p)$  can be employed as the readout train. We summarize the main steps of the Grassmann embedding in Algorithm 1.

#### 4.4. Stable SVD for Backward Propagation

While we package the embedding operations in an end-to-end learning framework, it is essential to develop a computational strategy for SVD that is reliable in back-propagation (BP) of deep neural networks. The rest of this section gives the derivation of BP for the employed truncated SVD, which is numerically stable especially in the case when the input matrix involves extremely small singular values. We denote two orthonormal matrices  $\mathbf{U} \in \mathbb{R}^{m \times k}$ ,  $\mathbf{V} \in \mathbb{R}^{n \times k}$ , and

$\mathbf{S} = \text{diag}(s_1, s_2, \dots, s_k) \in \mathbb{R}^{k \times k}$  as the output of SVD on  $H^\top$  during forward-propagation. To update  $H^\top$  in BP, its gradient is calculated by

$$\begin{aligned} \nabla_{H^\top} f = & \left[ \mathbf{U} \left( \mathbf{F} \circ \left[ \mathbf{U}^\top \bar{\mathbf{U}} - \bar{\mathbf{U}}^\top \mathbf{U} \right] \right) \mathbf{S} + \left( \mathbf{I}_m - \mathbf{U} \mathbf{U}^\top \right) \bar{\mathbf{U}} \mathbf{S}^{-1} \right] \mathbf{V}^\top \\ & + \mathbf{U} \left[ \mathbf{S} \left( \mathbf{F} \circ \left[ \mathbf{V}^\top \bar{\mathbf{V}} - \bar{\mathbf{V}}^\top \mathbf{V} \right] \right) \mathbf{V}^\top + \mathbf{S}^{-1} \bar{\mathbf{V}}^\top \left( \mathbf{I}_n - \mathbf{V} \mathbf{V}^\top \right) \right] \\ & + \mathbf{U} \left( \mathbf{I}_k \circ \bar{\mathbf{S}} \right) \mathbf{V}^\top, \end{aligned}$$

where  $\mathbf{F}_{ij} = \frac{1}{s_j^2 - s_i^2} \cdot \mathbb{1}\{i \neq j\}$  that satisfies the identity  $\mathbf{F}^\top = -\mathbf{F}$ . The calculations of  $\mathbf{F}$  and  $\mathbf{S}^{-1}$  are often numerically unstable due to the possible near-zero singular values. To circumvent this difficulty, we follow the authors of [56] and introduce the following trick on  $\mathbf{S}$ :

$$\mathbf{S}_{i,i}^{\text{new}} = \mathbf{S}_{i,i} \cdot \mathbb{1}\{\mathbf{S}_{i,i} > \epsilon\} + \epsilon \cdot \mathbb{1}\{\mathbf{S}_{i,i} \leq \epsilon\},$$

where  $\epsilon$  is a small number and can usually be set to  $10^{-12}$ . In practice, we replace  $\mathbf{S}$  by the modified matrix  $\mathbf{S}^{\text{new}}$  in BP to avoid 0 values in  $\mathbf{S}$ .

## 5. Applications on Graph

This section applies the proposed EGG to two distinct graph learning tasks: node clustering and graph classification. We start with formulating the two problems to be solved, following the designed model structure of the two addressed problems.

### 5.1. Graph Classification

A graph-level representation learning task, such as graph classification and regression, takes multiple graphs  $\mathbb{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_N\}$  as the input to train a feasible model that makes correct assignment  $\mathbf{y}_i = g(\mathbf{X}_i, \mathbf{A}_i)$ . The node sizes of different graphs are merely identical, and it is the duty of graph pooling to learn from a hidden graph representation  $\mathbf{H}_i = f_1(\mathbf{X}_i, \mathbf{A}_i)$  so that the graph is summarized to  $\mathbf{h}_i = f_2(\mathbf{H}_i)$  with a determined length. The representation is later employed for label prediction, i.e.,  $\mathbf{y}_i = f_3(\mathbf{h}_i)$ .

With GNNs, the first step of  $f_1(\cdot)$  is usually executed by graph convolutional layers to extract structure and node features, which outputs a hidden representation  $\mathbf{H} \in \mathbb{R}^{n \times m}$  for an arbitrary graph of  $n$  nodes. A graph pooling strategy is then selected to design a proper  $f_2(\cdot)$  that unifies the dimension of representations to all the graphs. The proposed EGG defines  $f_2(\cdot)$  by embedding each  $\mathbf{H}$  to a Grassmann point of  $\mathcal{M}(p, m)$ . Specifically, it calculates a  $k$ -dimensional subspace of  $\mathbf{H}^\top$  by the truncated SVD. The  $\mathbf{U}$  from (1) denotes a Grassmann point associated with a graph. To allow Euclidean computations, it can be projected to a symmetric matrix by (2) and be flattened and sent to a classifier, such as fully-connected layers. The embedding step is also illustrated in Figure 1 and Algorithm 1.

The validity of EGG for graph pooling is guaranteed by the two essential requirements: *uni-size representation* and *permutation invariance*, which we shall check now.

**Proposition 1.** EGG always produces a graph embedding  $g \in \mathbb{R}^{\frac{m(m+1)}{2}}$  for the node representation matrix  $\mathbf{H} \in \mathbb{R}^{n \times m}$ , regardless of the graph size  $n$ .

*Proof.* Given the node representation matrix  $\mathbf{H} \in \mathbb{R}^{n \times m}$  of a graph  $\mathcal{G}$  with  $n$  nodes and  $m$  features, the Grassmann graph embedding gives the output  $\mathbf{U}\mathbf{U}^\top$ , where  $\mathbf{U} \in \mathbb{R}^{m \times k}$  with  $k = \text{rank}(\mathbf{H}^\top)$ . Then, the output of EGG for the graph classification is the flattened representation of the upper triangular matrix of  $\mathbf{U}\mathbf{U}^\top \in \mathbb{R}^{m \times m}$ . The length of vector representation is thus  $m(m+1)/2$ , and it is independent of the graph size  $n$ .  $\square$

**Proposition 2.** EGG satisfies the requirement of permutation invariance so that it produces the same Grassmann graph embedding under row permutations of the input node representation matrix.

*Proof.* Suppose  $\mathbf{H}_1$  is the node representation matrix of a graph and let  $\mathbf{H}_2 = \mathbf{P}\mathbf{H}_1$ , where  $\mathbf{P}$  is a permutation matrix. Then,

$$\mathbf{H}_1^\top = \mathbf{U}\mathbf{S}\mathbf{V}^\top, \quad \mathbf{H}_2^\top = \mathbf{H}_1^\top \mathbf{P}^\top = \mathbf{U}\mathbf{S}\mathbf{V}^\top \mathbf{P}^\top.$$

The Grassmann point for both  $\mathbf{H}_1$  and  $\mathbf{H}_2$  can be accessed by the same matrix  $\mathbf{U}$ . Hence, the proposed graph embedding method is permutation invariant.  $\square$

## 5.2. Node Clustering

Node-level tasks make predictions on each node  $v$  of a single graph  $\mathcal{G}$ . For node clustering, the target is to segment the full graph to a number of subsets, where nodes from the same subset usually have closer connection to each other, and/or share similar properties.

A typical approach in literature to solve node clustering problems is by using *Variational Graph Auto-Encoder* (VGAE) [57] to generate a latent representation for a graph and then applying  $k$ -means [58] for clustering the nodes, where the VGAE model pursues the optimal variational parameters  $\mathbf{W}$  that minimize the variational lower bound

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X}, \mathbf{A})} [\log p(\mathbf{A}|\mathbf{Z})] - \text{KL}[q(\mathbf{Z}|\mathbf{X}, \mathbf{A}) \| p(\mathbf{Z})].$$

Here  $q(\mathbf{Z}|\mathbf{X}, \mathbf{A})$  is the encoder such that

$$q(\mathbf{Z}|\mathbf{X}, \mathbf{A}) = \prod_{i=1}^N q(\mathbf{z}_i|\mathbf{X}, \mathbf{A}), \quad q(\mathbf{z}_i|\mathbf{X}, \mathbf{A}) \sim \mathcal{N}(\mathbf{z}_i|\boldsymbol{\mu}_i, \text{diag}(\boldsymbol{\sigma}_i^2)).$$

Both the mean  $\boldsymbol{\mu}$  and log standard deviation  $\log \boldsymbol{\sigma}$  are approximated by graph convolutional layers, such as GCN [18]. The decoder is defined as

$$p(\mathbf{A}|\mathbf{Z}) = \prod_{i=1}^N \prod_{j=1}^N p(A_{ij}|\mathbf{z}_i, \mathbf{z}_j), \quad p(A_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^\top \mathbf{z}_j),$$

where  $\mathbf{A}$  is with respect to the adjacency matrix, and  $\sigma(\cdot)$  is the activation function. We take the inference set  $\mathbf{H} := [\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_m] \in \mathbb{R}^{n \times m}$  as the hidden representation of the graph with  $n$  nodes of hidden size  $m$  and send it to EGG for Grassmann embedding. The consequent  $\mathbf{U}\mathbf{U}^\top \in \mathbb{R}^{n \times n}$  is handled by clustering methods, such as  $k$ -means [58], to assign clusters with  $k$  set to be the number of classes of the dataset. Instead of relying on the pair-wise connection or the feature-space information, EGG leverages second-order correlations of nodes for a proper segmentation.

## 6. Experiment

This section evaluates the proposed framework on graph-level classification tasks as well as node-level clustering tasks. The former is conducted on six benchmarks of variant graph sizes, volume, and density, and the latter employs five popular graph datasets of moderate volume. All benchmark datasets and baseline methods are publicly available in the **PyTorch Geometric (PYG)** [59] library. The implementation of EGG is published at <https://github.com/conf20/Egg>. The rest of this section lists the experimental setup and analyzes the performance comparison of the two experiments.

### 6.1. Ablation Study on Graph Pooling

Table 1: Summary of the datasets for graph property prediction tasks.

Datasets	Proteins	D&D	NCI1	Mutagen	Collab	Molhiv
# graphs	1,113	1,178	4,110	4,337	5,000	41,127
# classes	2	2	2	2	3	2
Min # nodes	4	30	3	30	32	2
Max # Nodes	620	5,748	111	417	492	222
Avg # nodes	39	284	30	30	74	26
Avg # edges	73	716	32	31	2,458	28
# Features	3	89	37	14	0	9

#### 6.1.1. Setup

We benchmark the performance on five binary classification and one multi-class classification tasks. The **Molhiv** [60] is from open graph benchmark, and all other datasets are provided by **TUDataset benchmarks** [61]. The summary statistics of the six benchmark datasets is provided in Table 1. Most benchmark datasets preserve their original feature attributes except for the feature-less dataset **Collab**, where we follow [9] to generate new features with

Table 2: Performance comparison for graph classification with **GCN convolution**.

	<b>Proteins</b>	<b>D&amp;D</b>	<b>NCI1</b>	<b>Mutagen</b>	<b>Collab</b>	<b>Molhiv</b>
TopKPool	73.48 $\pm$ 3.57	74.87 $\pm$ 4.12	75.11 $\pm$ 3.45	79.84 $\pm$ 2.46	81.18 $\pm$ 0.89	77.11 $\pm$ 1.27
SAGPool	75.89 $\pm$ 2.91	74.96 $\pm$ 3.60	76.30 $\pm$ 1.53	79.86 $\pm$ 2.36	79.26 $\pm$ 5.37	75.36 $\pm$ 1.82
EDGEPool	75.60 $\pm$ 2.40	67.60 $\pm$ 0.51	77.17 $\pm$ 1.49	70.34 $\pm$ 1.69	75.09 $\pm$ 0.81	75.14 $\pm$ 1.66
PANPool	72.41 $\pm$ 3.58	72.52 $\pm$ 4.05	62.82 $\pm$ 3.04	70.14 $\pm$ 1.85	75.78 $\pm$ 2.03	74.18 $\pm$ 1.52
SUM	74.91 $\pm$ 4.08	78.91 $\pm$ 3.37	76.96 $\pm$ 1.70	80.69 $\pm$ 3.26	80.76 $\pm$ 1.56	74.88 $\pm$ 2.64
AVG	73.13 $\pm$ 3.18	76.89 $\pm$ 2.23	73.70 $\pm$ 2.55	80.37 $\pm$ 2.44	81.24 $\pm$ 1.34	<b>77.69</b> $\pm$ 1.17
MAX	73.57 $\pm$ 3.94	75.80 $\pm$ 4.11	75.96 $\pm$ 1.82	78.83 $\pm$ 1.70	82.28 $\pm$ 2.10	76.95 $\pm$ 0.94
Attention	73.93 $\pm$ 5.37	77.48 $\pm$ 2.65	74.04 $\pm$ 1.27	80.25 $\pm$ 2.22	81.58 $\pm$ 1.72	77.44 $\pm$ 1.27
EGG	<b>77.79</b> $\pm$ 2.16	<b>79.10</b> $\pm$ 2.98	<b>77.80</b> $\pm$ 2.01	<b>81.01</b> $\pm$ 1.28	<b>82.94</b> $\pm$ 1.06	76.60 $\pm$ 1.10

one-hot encoding of node degrees. Also, virtual nodes [62] are included in **Molhiv** to enhance the learning ability, as is suggested by the authors of [60].

To learn the hidden representation of graph topological embedding for pooling layers, we consider two variants of GCN [18] and GIN [9]. For the first five datasets from **TU Datasets**, we construct three GCN layers for **NCI1**, and two GCN layers for the other four datasets to encode graph hidden representation for pooling. Meanwhile, the number of GIN convolutional layers is set to four with the JKNET [22] construction. Both models for the largest dataset **Molhiv** use four convolutional layers. The learned hidden representation is sent to one of the baselines, following a two-layer MLP with size 64 and 16, respectively.

A fair comparison of EGG is made against four hierarchical and four global pooling methods. The former includes TOPKPOOL [30, 11], SAGPOOL [31] EDGEPOOL [63, 64], and PANPOOL [65], and the latter chooses ATTENTION [32], SUMMATION, AVERAGE and MAXIMIZATION methods. The official implementation are of all eight baselines are provided by PyTorch Geometric [59].

The models are trained on 80% randomly selected samples of the datasets, validated on 10% samples, and tested on the left 10% samples. The main hyper-parameters are fine-tuned with a grid search engine, where we are interested in



Table 3: Performance comparison for graph classification with **GIN convolution**.

	<b>Proteins</b>	<b>D&amp;D</b>	<b>NCI1</b>	<b>Mutagen</b>	<b>Collab</b>	<b>Molhiv</b>
TopKPool	73.66 $\pm$ 6.00	76.40 $\pm$ 2.32	77.06 $\pm$ 0.90	78.30 $\pm$ 1.36	81.40 $\pm$ 0.94	78.14 $\pm$ 0.62
SAGPool	75.95 $\pm$ 4.52	68.94 $\pm$ 7.62	76.97 $\pm$ 2.94	78.86 $\pm$ 1.58	81.76 $\pm$ 1.57	75.26 $\pm$ 2.29
EDGEPool	75.13 $\pm$ 3.62	72.82 $\pm$ 1.40	77.79 $\pm$ 2.80	81.01 $\pm$ 0.82	79.20 $\pm$ 1.66	75.30 $\pm$ 2.01
PANPool	71.43 $\pm$ 2.15	72.75 $\pm$ 2.32	71.68 $\pm$ 4.45	78.09 $\pm$ 1.27	80.22 $\pm$ 2.02	77.18 $\pm$ 1.13
SUM	78.04 $\pm$ 2.30	78.57 $\pm$ 1.26	78.83 $\pm$ 1.49	81.31 $\pm$ 1.10	82.64 $\pm$ 0.85	77.41 $\pm$ 1.16
AVG	71.70 $\pm$ 2.08	74.37 $\pm$ 1.32	76.55 $\pm$ 1.72	80.97 $\pm$ 1.18	<b>83.30</b> $\pm$ 0.77	<b>78.21</b> $\pm$ 0.90
MAX	76.70 $\pm$ 1.57	77.31 $\pm$ 2.06	79.27 $\pm$ 1.38	80.28 $\pm$ 0.83	80.94 $\pm$ 0.72	78.16 $\pm$ 1.33
Attention	75.63 $\pm$ 1.13	71.76 $\pm$ 3.26	78.22 $\pm$ 1.32	78.54 $\pm$ 5.37	83.22 $\pm$ 0.30	74.44 $\pm$ 2.12
EGG	<b>79.80</b> $\pm$ 1.09	<b>81.18</b> $\pm$ 1.14	<b>81.31</b> $\pm$ 1.55	<b>82.53</b> $\pm$ 0.72	81.32 $\pm$ 0.68	77.82 $\pm$ 0.90

Learning rate in  $\{5e-3, 1e-3, 5e-4\}$ ,  $L_2$  weight decay in  $\{5e-3, 1e-3, 5e-4\}$ , and hidden units in  $\{32, 64\}$  for the convolution layers. For the fully-connected layer, we search in particular the dropout ratio in  $\{0, 0.5\}$ . For EGG, we include an extra hyper-parameter of the threshold ratio  $r$  in truncated SVD, which is set to one of  $\{0.5, 0.8\}$ . The model stops training whenever the validation loss stops improving for 20 consecutive epochs, or the training epoch reaches 200.

### 6.1.2. Result

Table 2 and Table 3 compare the prediction performance of EGG with the eight baselines. Follow the convention, we report the percentage value of mean test accuracy for the classification tasks with **TUDatasets** and ROC-AUC score for **Molhiv**. The mean performance score are averaged over 10 repetitions with their standard deviations provided after the  $\pm$  signs. In general, our EGG achieves the top score on all the six tasks with a lower volatility. The advantage is more salient when the hidden representation of graphs are embedded by GIN convolutions with JKNET structure. EGG constructs a feasible global pooling method that interprets the second-order correlation of the compressed graph expressions, which are more informative than the first-order relationships. As this covariance relationship is captured by a non-linear transformation, i.e., a

Table 4: Summary of the datasets for node clustering tasks.

	<b>Cora</b>	<b>Citeseer</b>	<b>Pubmed</b>	<b>Wiki-CS</b>	<b>Coauthor-CS</b>
# Nodes	2,708	3,327	19,717	11,701	18,333
# Edges	5,429	4,732	44,338	216,123	100,227
# Features	1,433	3,703	500	300	6,805
# Clusters	7	6	3	10	15

truncated SVD, one layer of EGG is generally sufficient for graph distilling tasks, and it relieves the burden of the fine-tuning work in training a deep graph learning model. It should be emphasized that the correlation analysis of EGG relies on the node attributes. When the input graph is feature-less, such as **Collab**, the performance can be less-promising.

## 6.2. Node Clustering

### 6.2.1. Setup

The second experiment validates the design of EGG to node clustering tasks. Five popular benchmarks, including the three citation networks (**Cora**, **Citeseer**, **Pubmed**) [66], **Wiki-CS** [67] and **Coauthor-CS** [68], are employed to examine the effectiveness of an additional embedding step of EGG. The statistics of the five datasets are summarized in Table 4. In terms of the baseline method, we train a VGAE model [57] to generate the latent representation  $\mathbf{H} \in \mathbb{R}^{n \times m}$  of the graph and then send the  $\mathbf{H}$  matrix to  $k$ -means [58] for clustering. Based on this baseline architecture, we insert our EGG before the  $k$ -means as an additional step. Specifically, we first send the learned latent representation  $\mathbf{H}$  to EGG, then perform  $k$ -means on the output of the EGG procedure.

The same structure of the VGAE model is adopted from [57], where the encoder is implemented by a two-layer GCN [18] and the decoder is simply given by an inner product between the latent variables as illustrated in Section 5.2. We train the VGAE model for 200 epochs using ADAM [69] with a learning rate of 0.01. The dimensions of the hidden layer and the latent space are set to 32 and

Table 5: Performance comparison for node clustering. All scores are averaged over 10 repetitions with the scale between 0 and 1. The value after  $\pm$  is standard deviation.

	Method	Acc.	NMI	ARI	CS
Cora	$k$ -means	0.6137 $\pm$ 0.0345	0.4459 $\pm$ 0.0190	0.3782 $\pm$ 0.0312	0.4351 $\pm$ 0.0188
	$k$ -means+EGG0.2 (53.16%)	0.5207 $\pm$ 0.0553	0.3789 $\pm$ 0.0374	0.2967 $\pm$ 0.0475	0.3702 $\pm$ 0.0360
	$k$ -means+EGG0.5 (90.73%)	0.6195 $\pm$ 0.0340	0.4345 $\pm$ 0.0269	0.3804 $\pm$ 0.0434	0.4292 $\pm$ 0.0257
	$k$ -means+EGG0.8 (98.76%)	<b>0.6388</b> $\pm$ 0.0386	<b>0.4548</b> $\pm$ 0.0158	<b>0.3998</b> $\pm$ 0.0293	<b>0.4591</b> $\pm$ 0.0191
Citeseer	$k$ -means	0.4347 $\pm$ 0.0341	0.1931 $\pm$ 0.0343	0.1548 $\pm$ 0.0262	0.1942 $\pm$ 0.0358
	$k$ -means+EGG0.2 (45.28%)	0.4156 $\pm$ 0.0409	0.1794 $\pm$ 0.0259	0.1527 $\pm$ 0.0294	0.1783 $\pm$ 0.0254
	$k$ -means+EGG0.5 (88.25%)	0.4683 $\pm$ 0.0323	<b>0.2098</b> $\pm$ 0.0283	<b>0.1865</b> $\pm$ 0.0301	0.2096 $\pm$ 0.0282
	$k$ -means+EGG0.8 (97.98%)	<b>0.4698</b> $\pm$ 0.0284	0.2071 $\pm$ 0.0235	0.1859 $\pm$ 0.0282	<b>0.2107</b> $\pm$ 0.0201
Pubmed	$k$ -means	0.6469 $\pm$ 0.0232	0.2425 $\pm$ 0.0233	0.2380 $\pm$ 0.0368	0.2393 $\pm$ 0.0233
	$k$ -means+EGG0.2 (64.98%)	0.6339 $\pm$ 0.0193	0.2311 $\pm$ 0.0162	0.2196 $\pm$ 0.0228	0.2280 $\pm$ 0.0163
	$k$ -means+EGG0.5 (97.18%)	0.6296 $\pm$ 0.0228	0.2394 $\pm$ 0.0252	0.2171 $\pm$ 0.0302	0.2377 $\pm$ 0.0263
	$k$ -means+EGG0.8 (99.48%)	<b>0.6521</b> $\pm$ 0.0135	<b>0.2532</b> $\pm$ 0.0201	<b>0.2436</b> $\pm$ 0.0231	<b>0.2521</b> $\pm$ 0.0224
Wiki-CS	$k$ -means	0.4080 $\pm$ 0.0371	0.3429 $\pm$ 0.0157	0.2156 $\pm$ 0.0240	0.3418 $\pm$ 0.0153
	$k$ -means+EGG0.2 (52.48%)	0.3875 $\pm$ 0.0392	0.3207 $\pm$ 0.0183	0.2065 $\pm$ 0.0276	0.3195 $\pm$ 0.0179
	$k$ -means+EGG0.5 (93.21%)	0.4103 $\pm$ 0.0253	0.3319 $\pm$ 0.0132	0.2232 $\pm$ 0.0249	0.3307 $\pm$ 0.0128
	$k$ -means+EGG0.8 (99.43%)	<b>0.4685</b> $\pm$ 0.0210	<b>0.3643</b> $\pm$ 0.0193	<b>0.2423</b> $\pm$ 0.0223	<b>0.3633</b> $\pm$ 0.0181
Coauthor-CS	$k$ -means	0.6410 $\pm$ 0.0171	0.6950 $\pm$ 0.0311	0.5442 $\pm$ 0.0427	0.6880 $\pm$ 0.0300
	$k$ -means+EGG0.2 (45.05%)	0.5483 $\pm$ 0.0250	0.6095 $\pm$ 0.0210	0.4472 $\pm$ 0.0269	0.6050 $\pm$ 0.0235
	$k$ -means+EGG0.5 (92.67%)	0.5878 $\pm$ 0.0188	0.6750 $\pm$ 0.0150	0.5033 $\pm$ 0.0190	0.6735 $\pm$ 0.0157
	$k$ -means+EGG0.8 (99.68%)	<b>0.6700</b> $\pm$ 0.0201	<b>0.7248</b> $\pm$ 0.0195	<b>0.6150</b> $\pm$ 0.0205	<b>0.7230</b> $\pm$ 0.0193

16 in all experiments, respectively. To obtain the latent representation of nodes, VGAE is trained on a link prediction task of identifying edges and non-edges. Following [57], we divide the dataset into the training, validation and test sets by a random selection of 85%, 5% and 10% of the total node connections. The positive and negative edges share the same amount. In terms of evaluation, we employ the following four metrics to validate the clustering results: Accuracy (Acc.), Normalized Mutual Information (NMI), Average Rand Index (ARI), and Completeness Score (CS).

### 6.2.2. Result

Table 5 reports the experimental results of EGG-enhanced  $k$ -means with VGAE in node clustering tasks. In addition to the four evaluation metrics, we also attach two additional ratios in the name of  $k$ -means+EGG $_x(y)$ , where the first ratio  $x \in (0, 1)$  is used to determine the number of most important components  $p$  in the latent space of  $\mathbf{H} \in \mathbb{R}^{n \times m}$  we have kept, that is  $p = \lceil xm \rceil$ . The second value  $y$  indicates the percentage of the information from the latent representation  $\mathbf{H}$  being captured by these  $p$  components.

It shows clearly from Table 5 that the EGG0.8-enhanced  $k$ -means consistently outperforms the plain  $k$ -means on all the datasets in terms of all four metrics. We also observe that EGG0.5 provide a comparable performance against the baseline, giving that the first  $p = \lceil 0.5m \rceil$  components contains 90% variations of the graphs’ latent representation. These observations confirm that EGG brings solid performance gains to the node clustering tasks.

## 7. Further Investigation

This section validates EGG from three perspectives. The sensitivity of the model performance to the newly introduced hyperparameter is tested in the first part. We then explore the expressiveness of the learned embedding with a t-SNE visualization of the learned hidden representation. In the last sector, we check the learning behavior of EGG through the loss curve of training tasks.

### 7.1. Sensitivity to the embedded dimension

As discussed in Section 4, the number of the subspace dimension  $p$  is adaptively selected for each Grassmann point, according to the global threshold of the percentage importance  $r$ . The  $r$  is thus a new hyperparameter for EGG. This section designs a sensitivity analysis to demonstrate the negligible impact of the choice of  $r$  on the performance of the Grassmann embedding. The model learns a graph classification task on **Proteins** with both GCN and GIN networks, which architectures and training setups are detailed in Section 6.1. We

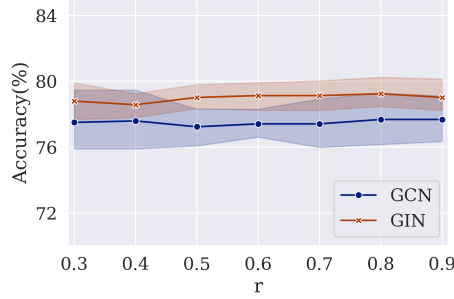


Figure 2: Sensitivity analysis for the threshold information ratio  $r$  on **PROTEINS**.

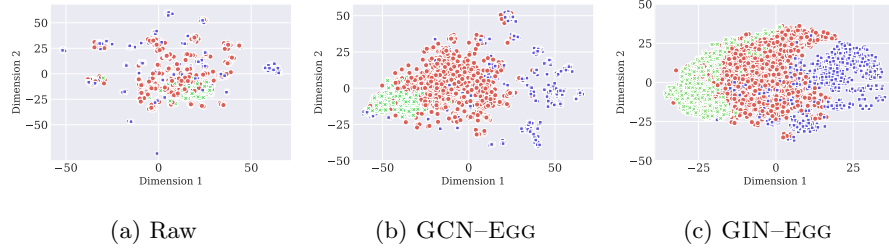


Figure 3: The t-SNE visualizations of graph representations produced by EGG pooling on **COLLAB** with GCN before training; with GCN after training; and with GIN after training.

report the mean test accuracy over 10 repetitive runs. Seven different values of the threshold ratio  $r$  is set from 0.3 to 0.9 with step size 0.1. The results are visualized in Figure 2, which draws a nearly horizontal trend of the mean test accuracy movement of EGG with different choices of the threshold ratio  $r$ . This suggests that the considerably wide choice of the hyperparameter  $r$  does not drastically influence the performance of EGG pooling. In fact, we suggest a moderately high value of  $r$ , such as 0.5, to retain the essential information of a graph in Grassmann embedding and guarantee a relatively fast computational speed of the algorithm at the same time.

## 7.2. Embedding Expressiveness

Next, we exploit the expressiveness of the flattened Euclidean graph embeddings with two-dimensional t-distributed Stochastic Neighbor Embedding

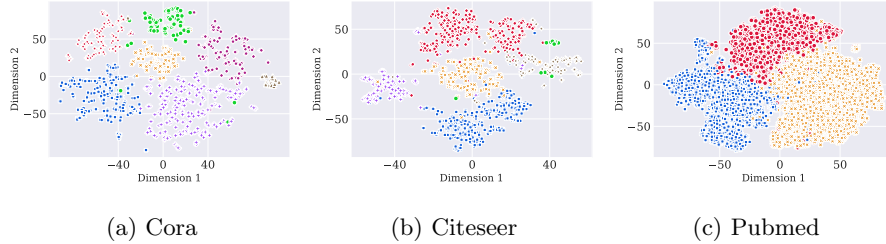


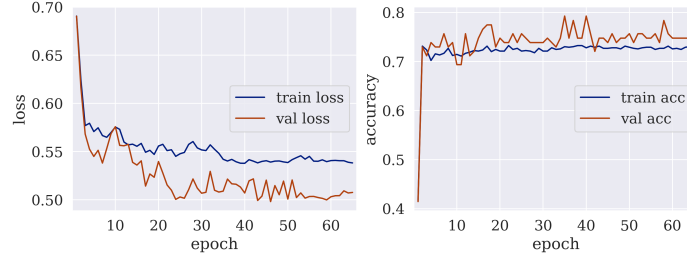
Figure 4: The t-SNE visualizations of node representations with EGG(0.8)+ $k$ -means on **Cora**, **Citeseer** and **Pubmed**.

(t-SNE). The results are from the 3-class graph classification task **Collab** that we conducted in the first experiment of Section 6. In Figure 3, each point denotes a graph hidden representation by EGG, and the three colors indicate one of the three true labels. For a more clear presentation, we sample 4,000 instances in random. In the case of the GIN convolution, outputs from all the four pooling layers are aggregated, due to the employed JKNET structure. Both Figures 3(b) and 3(c) suggest a clear clustering pattern of the pooled graphs.

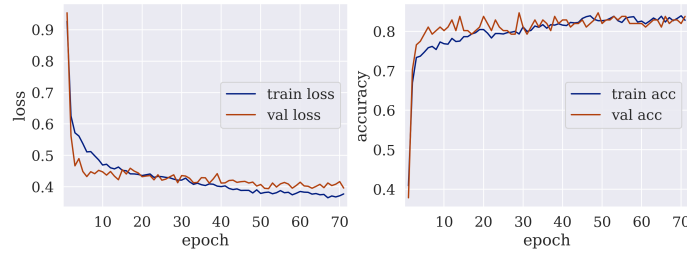
A similar visualization for the learned representation in node clustering tasks is displayed in Figure 4, where the three citation networks are trained. Once again, points of different colors are basically located at distinct corners, which implies that the hidden representations from EGG(0.8)+ $k$ -means manage to collect a differentiable pattern for the underlying clustering task.

### 7.3. Learning Behaviors

In the last investigation, we check the training and validation curves for loss and accuracy in Figure 5. The results are retrieved from the graph classification learning task on **Proteins** with the same experimental settings in Section 6.1. Here only a single run results are retrieved due to the employment of the early stopping criteria, with which every independent run could stop at a different epoch. Except for the minor volatility after epoch 5 that is very likely brought about by stochastic gradients, all the four training lines validate an efficient convergence of EGG, where the loss curve stabilizes quickly after a few epochs.



(a) GCN Convolution



(b) GIN Convolution

Figure 5: Training and Validation learning curves on the **Proteins** dataset with EGG pooling. The graph convolutional layers are set to GCN and GIN, respectively.

The training process is slightly longer for GIN convolution, which is partly due to the more sophisticated network architecture for the model to fit.

## 8. Discussion and Conclusion

This paper develops a Grassmann geometry-based graph embedding strategy named EGG. For a given set of hidden feature subspace of graphs, the proposed method rectifies them to Grassmann points of a Grassmann manifold to make analysis on them. Through establishing the view of treating graph nodes as a subspace, many new perspectives on formulating informative graph representations become visible. For example, this work approximates the covariance relationship of node attributes with non-linearity transformation, which concurrently offsets the inefficiency of a stack of fully-connected layers and blurs the minor perturbations of the initial representation. Furthermore, the new

framework allows a swift projection from the manifold space back to the Euclidean space, so that the new representation supports common loss designs by Euclidean metrics. We demonstrate effectiveness of the embedding framework with extensive numerical experiments, for both graph-level and node-level representation learning tasks.

The proposed EGG has multiple aspects of potential. As said, it defines a non-linear transformation routine to graph features, which can be exceptional when working on complex or massive attributes. Moreover, treating regional graphs as Grassmann points or other entities from non-Euclidean space brings extra flexibility to the model design. We believe this idea could motivate more links of graph topological learning and other geometric learning schemes.

## References

- [1] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: going beyond euclidean data, *IEEE Signal Processing Magazine* 34 (4) (2017) 18–42.
- [2] W. L. Hamilton, *Graph Representation Learning*, Morgan & Claypool Publishers, 2020.
- [3] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip, A comprehensive survey on graph neural networks, *IEEE Transactions on Neural Networks and Learning Systems* 32 (1) (2020) 4–24.
- [4] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: a review of methods and applications, *AI Open* 1 (2020) 57–81.
- [5] Z. Zhang, P. Cui, W. Zhu, Deep learning on graphs: a survey, *IEEE Transactions on Knowledge and Data Engineering* (2020).
- [6] J. Bruna, W. Zaremba, A. Szlam, Y. Lecun, Spectral networks and locally connected networks on graphs, in: *ICLR*, 2014.



- [7] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, Neural message passing for quantum chemistry, in: NeurIPS, Vol. 70, 2017, pp. 1263–1272.
- [8] Z. Wang, S. Ji, Second-order pooling for graph neural networks, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020).
- [9] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, in: ICLR, 2019.
- [10] X. Zheng, B. Zhou, J. Gao, Y. G. Wang, P. Liò, M. Li, G. Montúfar, How framelets enhance graph neural networks, ICML (2021).
- [11] H. Gao, S. Ji, Graph U-nets, in: ICML, 2019.
- [12] X. Zheng, B. Zhou, M. Li, Y. G. Wang, J. Gao, MathNet: Haar-like wavelet multiresolution-analysis for graph representation and learning, arXiv:2007.11202 (2020).
- [13] Y. Ma, S. Wang, C. C. Aggarwal, J. Tang, Graph convolutional networks with eigenpooling, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 723–731.
- [14] S. Kolouri, N. Naderializadeh, G. K. Rohde, H. Hoffmann, Wasserstein embedding for graph learning, in: ICLR, 2021.
- [15] F. Mémoli, Gromov-Wasserstein distances and the metric approach to object matching, Foundations of Computational Mathematics 11 (2011) 417–487.
- [16] P.-A. Absil, R. Mahony, R. Sepulchre, Optimization algorithms on matrix manifolds, Princeton University Press, 2008.
- [17] B. Zhou, X. Zheng, Y. G. Wang, M. Li, J. Gao, Grassmann graph embedding, in: ICLR 2021 Workshop on Geometrical and Topological Representation Learning, 2021.

- [18] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: ICLR, 2017.
- [19] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn: Simplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 639–648.
- [20] D. Bo, X. Wang, C. Shi, H. Shen, Beyond low-frequency information in graph convolutional networks, in: AAAI, Vol. 35, 2021, pp. 3950–3957.
- [21] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: NIPS, 2017.
- [22] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, S. Jegelka, Representation learning on graphs with jumping knowledge networks, in: ICML, 2018.
- [23] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: ICLR, 2018.
- [24] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P. S. Yu, Heterogeneous graph attention network, in: WWW, 2019, pp. 2022–2032.
- [25] D. Kim, A. Oh, How to find your friendly neighborhood: Graph attention design with self-supervision, in: ICLR, 2021.
- [26] M. Brockschmidt, Gnn-film: Graph neural networks with feature-wise linear modulation, in: ICML, 2020, pp. 1144–1152.
- [27] S. A. Taylor, F. Opolka, P. Lio, N. D. Lane, Do we need anisotropic graph neural networks?, in: ICLR, 2021.
- [28] T.-Y. Lin, A. RoyChowdhury, S. Maji, Bilinear cnn models for fine-grained visual recognition, in: ICCV, 2015, pp. 1449–1457.

- [29] Y. Wang, M. Long, J. Wang, P. S. Yu, Spatiotemporal pyramid network for video action recognition, in: CVPR, 2017, pp. 1529–1538.
- [30] C. Cătălina, P. Veličković, N. Jovanović, T. Kipf, P. Liò, Towards sparse hierarchical graph classifiers, in: NeurIPS Workshop on Relational Representation Learning, 2018.
- [31] J. Lee, I. Lee, J. Kang, Self-attention graph pooling, in: ICML, 2019.
- [32] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, in: ICLR, 2016.
- [33] B. Knyazev, G. W. Taylor, M. Amer, Understanding attention and generalization in graph neural networks, in: NeurIPS, Vol. 32, 2019.
- [34] D. Mesquita, A. H. Souza, S. Kaski, Rethinking pooling in graph neural networks, in: NeurIPS, 2020.
- [35] W. Dai, E. Kerman, O. Milenkovic, A geometric approach to low-rank matrix completion, IEEE Transactions on Information Theory 58 (1) (2012) 237–247.
- [36] N. Boumal, P.-A. Absil, Low-rank matrix completion via preconditioned optimization on the Grassmann manifold, Linear Algebra and its Applications 475 (2015) 200–239.
- [37] Y. M. Lui, Advances in matrix manifolds for computer vision, Image and Vision Computing 30 (6-7) (2012) 380–388.
- [38] H. Q. Minh, V. Murino, H. Q. Minh, Algorithmic advances in Riemannian geometry and applications, Springer, 2016.
- [39] Z. Huang, R. Wang, S. Shan, X. Chen, Projection metric learning on Grassmann manifold with application to video based face recognition, in: CVPR, 2015, pp. 140–149.

- [40] R. Slama, H. Wannous, M. Daoudi, A. Srivastava, Accurate 3d action recognition using learning on the Grassmann manifold, *Pattern Recognition* 48 (2) (2015) 556–567.
- [41] O. Koch, C. Lubich, Dynamical low-rank approximation, *SIAM Journal on Matrix Analysis and Applications* 29 (2) (2007) 434–454.
- [42] T. Ngo, Y. Saad, Scaled gradients on Grassmann manifolds for matrix completion, in: *NIPS*, 2012, pp. 1412–1420.
- [43] X. Dong, P. Frossard, P. Vandergheynst, N. Nefedov, Clustering on multi-layer graphs via subspace analysis on Grassmann manifolds, *IEEE Transactions on Signal Processing* 62 (4) (2014) 905–918.
- [44] B. Zhou, J. Gao, M.-N. Tran, R. Gerlach, Manifold optimization-assisted gaussian variational approximation, *Journal of Computational and Graphical Statistics* 30 (4) (2021) 946–957.
- [45] A. Edelman, T. A. Arias, S. T. Smith, The geometry of algorithms with orthogonality constraints, *SIAM journal on Matrix Analysis and Applications* 20 (2) (1998) 303–353.
- [46] P.-A. Absil, R. Mahony, R. Sepulchre, Riemannian geometry of Grassmann manifolds with a view on algorithmic computation, *Acta Applicandae Mathematica* 80 (2) (2004) 199–220.
- [47] Z. Huang, J. Wu, L. Van Gool, Building deep networks on Grassmann manifolds, in: *AAAI*, Vol. 32, 2018.
- [48] B. Wang, Y. Hu, J. Gao, Y. Sun, B. Yin, Low rank representation on Grassmann manifolds, in: *Asian Conference on Computer Vision (ACCV)*, 2014.
- [49] B. Wang, Y. Hu, J. Gao, Y. Sun, H. Chen, M. Ali, B. Yin, Locality preserving projections for Grassmann manifold, in: *IJCAI*, 2017, pp. 2893–2900.

- [50] T. Bendokat, R. Zimmermann, P.-A. Absil, A Grassmann manifold handbook: Basic geometry and computational aspects, arXiv:2011.13699 (2020).
- [51] K. A. Gallivan, A. Srivastava, X. Liu, P. Van Dooren, Efficient algorithms for inferences on Grassmann manifolds, in: IEEE Workshop on Statistical Signal Processing, 2003, IEEE, 2003, pp. 315–318.
- [52] Y. Chikuse, Statistics on Special Manifolds, Vol. 174, Springer Science & Business Media, 2003.
- [53] K. Ye, L.-H. Lim, Schubert varieties and distances between subspaces of different dimensions, SIAM Journal on Matrix Analysis and Applications 37 (3) (2016) 1176–1197.
- [54] M. T. Harandi, M. Salzmann, S. Jayasumana, R. Hartley, H. Li, Expanding the family of Grassmannian kernels: An embedding perspective, in: ECCV, Springer, 2014, pp. 408–423.
- [55] O. Tuzel, F. Porikli, P. Meer, Region covariance: A fast descriptor for detection and classification, in: ECCV, Springer, 2006, pp. 589–600.
- [56] Z. Huang, L. Van Gool, A Riemannian network for SPD matrix learning, in: AAAI, Vol. 31, 2017.
- [57] T. N. Kipf, M. Welling, Variational graph auto-encoders, in: NIPS Workshop on Bayesian Deep Learning, 2016.
- [58] S. Lloyd, Least squares quantization in pcm, IEEE Transactions on Information Theory 28 (2) (1982) 129–137.
- [59] M. Fey, J. E. Lenssen, Fast graph representation learning with PyTorch Geometric, in: ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.
- [60] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, J. Leskovec, Open graph benchmark: Datasets for machine learning on graphs, in: NeurIPS, 2020.

- [61] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, M. Neumann, TUDataset: A collection of benchmark datasets for learning with graphs, in: ICML Workshop on Graph Representation Learning and Beyond, 2020.
- [62] K. Ishiguro, S.-i. Maeda, M. Koyama, Graph warp module: an auxiliary module for boosting the power of graph neural networks, arXiv:1902.01020 (2019).
- [63] F. Diehl, T. Brunner, M. T. Le, A. Knoll, Towards graph pooling by edge contraction, in: ICML 2019 Workshop on Learning and Reasoning with Graph-structured Data, 2019.
- [64] F. Diehl, Edge contraction pooling for graph neural networks, arXiv:1905.10990 (2019).
- [65] Z. Ma, J. Xuan, Y. G. Wang, M. Li, P. Liò, Path integral based convolution and pooling for graph neural networks, in: NeurIPS, Vol. 33, 2020, pp. 16421–16433.
- [66] Z. Yang, W. Cohen, R. Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in: ICML, 2016.
- [67] P. Mernyei, C. Cangea, Wiki-CS: a wikipedia-based benchmark for graph neural networks, in: ICML Workshop on Graph Representation Learning and Beyond workshop, 2020.
- [68] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, in: NeurIPS Workshop on Relational Representation Learning Workshop, 2018.
- [69] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR, 2015.